Final Project

DEGREE IN

INFORMATICS ENGINEERING

PATIENT-PREOPERATORY IMAGE REGISTRATION FOR ROBOT ASSISTED SURGERY

Barcelona, 26 June 2013

Autor: Valentina Mura
Director: Alícia Casals Gelpí
Co-directors: Eduard Bergés Martín
and Cecilia Di Ruberto

ABSTRACT

This project is focused on the context of robot calibration for its application to surgery. The laboratory in which this work has been developed disposes of two robotic arms used for experimental activities, which frequently require calibrating their position. Calibrating the position of two robots, one with respect to the other, consists of finding their spatial relationship in a common coordinate system.

The objective of this project is to calibrate the position of robots by using a vision system. A calibration system was already used in the laboratory, and consists of a manual procedure. Since manually positioning of the robot arms is a long and tedious operation, it has been clear the need of having a faster and more user friendly process by using a vision system. In order to calibrate the robots it is necessary to calibrate the camera, finding its internal features and its position with respect to each robot.

The project has required the study of the theoretical bases about camera calibration and some techniques found in literature to achieve it. A known and reliable calibration methodology by vision is considered as reference method and implemented. The reference method requires computing the internal features of the camera in order to find its position with respect to the robots. In order to avoid the intrinsic calibration of the camera, an alternative methodology is designed and developed. This method uses a purely geometric procedure. It should allow carrying out the calibration process in a faster and more efficient way than the reference method, with at least the same accuracy.

In order to validate the results of the calibration, the accuracies obtained with both methods have been calculated and compared.

INDEX

1. INTRODUCTION

The use of robotic systems has considerably increased in recent decades. The development of more and more sophisticated devices has improved their use in different areas, such as industry, aviation and medicine.

In the context in which this project is focused, that is surgery, robotics can play an important role to satisfy the needs that may arise in a surgical procedure. The aim of surgical robots does not consist of substituting the surgeons, but provide additional features to improve the abilities of traditional surgery (for example introducing virtual fixtures, scaled movement or tremor reduction), typically characterized by manual operations. The laboratory in which this work has been developed disposes of three robotic arms.

Robotic surgery may have crucial contributions in the medical field. Due to the critical risks associated to a surgical procedure, it is necessary to ensure maximum accuracy and reliability of any device involved, i.e. the robotic arms used in a laparoscopic surgery. It is essential to know in each moment the precise position of the robot with respect to the working environment.

Often, robotic surgeries are previously planned by means of preoperative images. In order to relate the images with the surgical plan and the robots in the real environment, it is necessary to perform a registration process. The registration process consists of determining, through a set of preoperative images, the spatial relationship of some coordinate systems: that of the robot, that of the vision system and that of an anatomic object in the robot's work space. When registration is complete, a designated position on the preoperative image can be translated into a particular configuration of robot’s geometric parameters. That configuration will make the robot put a surgical tool at the position previously designated in the image.

Calibration is one of the processes required in the global registration process, although it may be necessary for other purposes. By means of calibration, it is possible to determine the position and orientation of the robot coordinates system, with respect to another coordinate system that is usually called “world coordinate system” (which can be the frame of another robot, the frame of a camera or a fix point of the laboratory, etc.).

In order to calibrate the robots, a video camera is used in this project. This choice, however, involves some problems that are inherent to vision. The hardest to calculate are the camera intrinsic characteristics (which will be described in 3.3). Therefore, a method which avoids this step would significantly decrease time and efforts to complete the whole process.

1.1. Motivation

Calibration is a fundamental process for any activity executed in the laboratory, which involves moving the robots. Calibrating the position of two robots, one with respect to the other, consists of finding their spatial relationship in a common coordinate system. As a result, any 3D position can be referred to robots' own coordinate systems and their kinematics can be related one with respect to the other. In any application where the robots share the workspace, this control is necessary in order to coordinate their movements (i.e. collision avoidance, cooperative tasks, remote guidance, etc.).

The main drawbacks of manual calibration process are:

- **Time consuming**: If there is no a calibration automatic process, calibrating a robot may take a long time.
- **Lack of accuracy**: Manual operations can introduce some errors regarding accuracy. Any mistake made during calibration could bring significant inconsistencies between the desired movements and the effective movements of the robot arms.

One of the most common methods to calibrate the position of the robots is through vision. The principal advantages of this solution are its immediacy and the possibility of developing simple interfaces that can use techniques of image processing, like segmentation, in order to obtain spatial information.

On the other hand, vision systems also involve some drawbacks. Sometimes it's necessary to know the internal parameters of the camera for a proper image data processing. For example, the lens distortion can alter the geometric relationships between the objects shown in the image and its influence should be considered in the calibration method. Also the position on which the camera is placed for the acquisition of the images has to be parameterized. Therefore, if the camera moves or any of its optical characteristics are modified, it's necessary to recalibrate the camera before starting any application that relies on information from its images.

Among the activities carried out in the laboratory, the use of robot arms instead of manual tools in surgery is experimented. These activities frequently require calibrating the position of the robots. The calibration system used in the laboratory consists of positioning both robots manually on some known positions of a pattern, which works as reference frame. Having the coordinates of these positions with respect to both robots' reference frames, a linear system is solved to find the relationship between the robots coordinates system.

Since manually positioning of the robot arms is a long and tedious operation, it is clear the need of having a faster and more user friendly process by using a vision system, which provides at least the same accuracy and reliability of the existing system. In fact, for the activities carried out in the laboratory, any improvement with respect to the calibration procedures actually adopted would be very useful.

In Fig. 1.1, the work environment, in the context of robot calibration by vision, is shown.

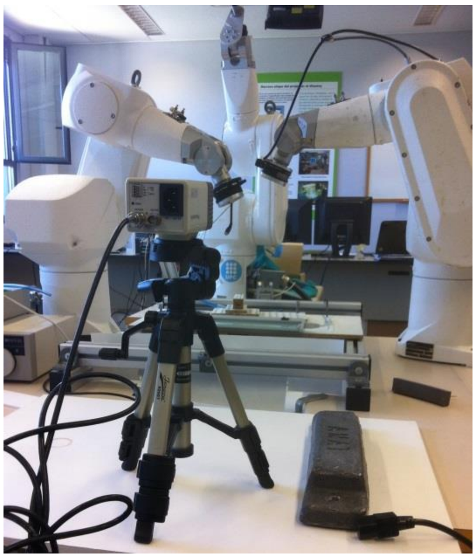


Figure 1.1 – Robot experimental platform and the camera for robots calibration

1.2. Objectives

The objective of this project is to calibrate the position of robots available in the laboratory by using a vision system. A known calibration methodology by vision will be considered as reference method and implemented (Chapter 4) by means of a MATLAB tool and Visual Studio. It will be compared to an alternative methodology (described in Chapter 5) that is expected to bring improvements regarding the main calibration drawbacks above described, without jeopardizing the accuracy. The proposed alternative method should verify the following constraints:

- the robot calibration is based on vision, using a single camera,
- the process is easier and faster than the reference method, with an acceptable degree of position accuracy (< 2,5mm)
- the position and orientation of the camera with respect to the robots is initially unknown
- the process is done without considering the internal features of the camera (avoidance of intrinsic calibration).

The implementation of the proposed alternative method, with the achievement of the declared purposes, should allow carrying out the activities of the laboratory, in which it is necessary that the robots work in a coordinate way, but through a faster and more efficient calibration process.

1.3. Work description

In order to understand the concepts used along the thesis, a description of the theoretical bases about the calibration of a camera are introduced in Chapter 3.

Then, after testing in the laboratory a known methodology for the camera calibration, that in this project is called “reference method” (Chapter 4), the work consists of reviewing a set of existing calibration techniques in order to evaluate which procedure can be implemented as the best fitting solution (Chapter 5). As explained above, the proposed alternative method has been designed so that it respects the constraints listed in the previous paragraph and satisfies the objectives declared in the same.

In order to validate the results of the calibration, the accuracy obtained with both methods has been calculated and compared (Chapter 7).

The work has required carrying out the following activities:

- Study of **theoretical basis** about camera calibration and some techniques found in literature to achieve it.
- Manufacturing of an infrared end effector in order to detect the positions of the robot with respect to the camera.
- Development of an interface to obtain and register the information necessary to implement the camera calibration. This interface is used for both the reference and the alternative methods.
- Design of the methodology named **“reference method”,** selecting the software packages in order to implement it.
    - Computation of the internal features of the camera, which are required in the reference method, by means of a MATLAB tool (Chapter 4),
    - Development of an interface that allows making the extrinsic calibration of the camera
- Design and solving, by means of MATLAB, of an **alternative method** which is expected to obtain at least the same accuracy results than the reference method.

- **Accuracy test design**, to determine whether the alternative method provides satisfactory results (Chapter 7).

1.4. Prerequisites

In order to understand the arguments treated in this project, some theoretical concepts related to computer vision and robotics are required. Practical knowledge about programming allowed the implementation of the interfaces described in Chapters 4 and 6.

Theoretical concepts:

- **Linear algebra and geometry**, since the work is heavily based on the use of matrices and vectors, on solving linear and non-linear systems, and, particularly for the alternative method, on 3D geometry (lines, planes, etc…).
- **Image processing**, useful for the interface that acquires 3D points. In fact, as described in Chapter 6, the image acquired from the camera undergoes some operations in order to extract the exact point that is reached by the robotic arm.
- **Computer vision,** since the camera calibration is based on its intrinsic and extrinsic parameters (that are explained in Chapter 3).
- **Robotics**, since it is necessary to use the transformation matrices in order to make the changes of the coordinate systems.

Practical knowledge:

- **MATLAB** (scripts and functions), in order to use the Camera Calibration Tool and to solve the equations defined in the alternative method
- **C++** programming on **Visual Studio C++** environment, in order to implement the interfaces for acquiring the points and calibrating the camera in the reference method
- **OpenCv** libraries, that are necessary for the operations of image processing and camera calibration

Physical requirements:

- A digital **video camera** connected to a PC by a video card
- A **PC** with the software for Windows that is mentioned above
- Two **robotic arms** used to acquire the points and to test the system
- A **control unit** that allows PC-robot communication via Ethernet

2. DESCRIPTION OF WORK ENVIRONMENT

In this chapter the equipment used to develop this project is described. The elements needed in order to carry out the work are:

- two robotic arms, which are the elements to be localized, one with respect to the other one,
- a camera, which is used to acquire the images used to calibrate robots by vision
- a CPU connected to robots, used to control the robots movements,
- a PC with Windows, Visual Studio, OpenCV and MATLAB, used to develop the software and to communicate with CPU and camera.

As shown in Fig. 2.7, the user can control the entire system by using the PC, which is directly connected to the video camera via a graphics card. The PC is connected to the robots via Ethernet through the CPU. Moreover, the processor has a manual control panel (MCP), physically connected to its control unit. Each element is described in the following paragraphs.

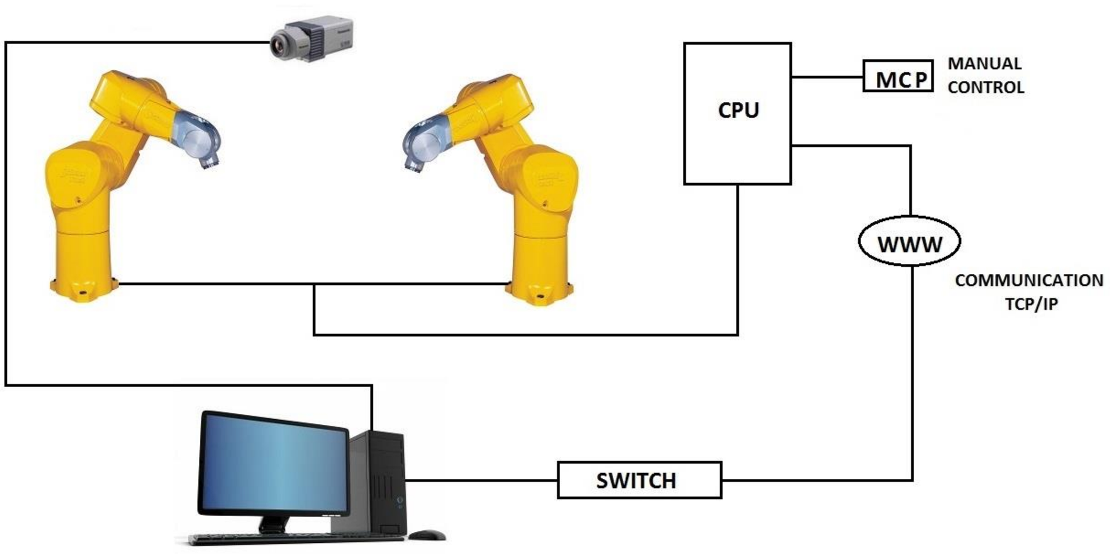


Figure 2.1 - Scheme of the work setup

2.1. Robot

Two robot arms branded Stäubli (model RX60 and RX60B). Each arm consists of segments or elements connected by joints (numbered from 1 to 6 in fig 2.1).

The movements of the joints are generated by coupled actuators. As seen in Fig. 2.2, the elements that compose the robot are: foot (A), shoulder (B), arm (C), elbow (D), forearm (E) and wrist (F).

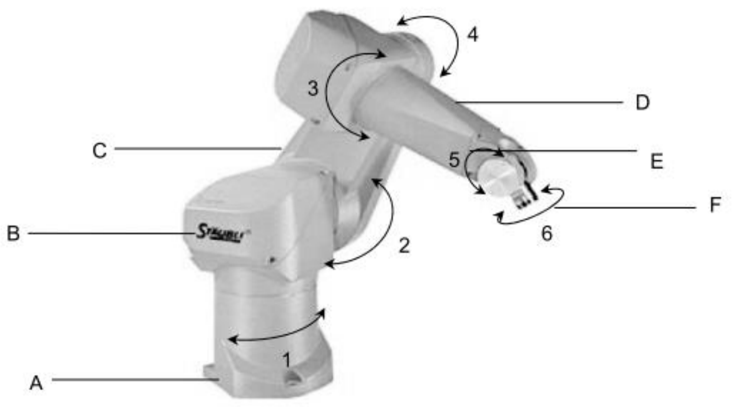


Figure 2.2 - The robot Stäubli with the enumeration of its segments and joints

At the end of arm, it is possible to connect different types of tools (end effectors), depending on the expected function to be performed by the arm (forceps, clamps, etc…). In order to acquire the projections of 3D points on the image, the end effector of both robots consists of a nylon stick with an infrared LED (Fig. 2.4). The end effectors used for accuracy testing are sharp tips (Fig. 2.3).

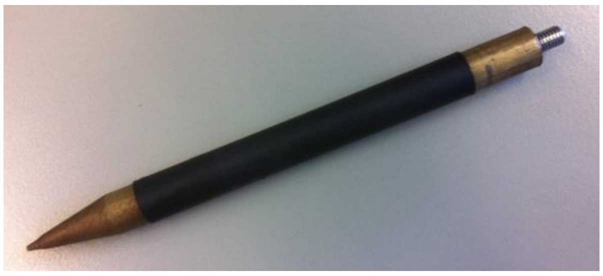


Figure 2.3 - The sharp tip end effector

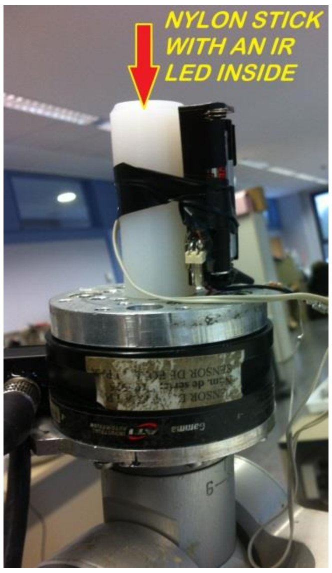


Figure 2.4 - The infrared LED placed as end effector

2.2. Controller

It is the unit that controls the movements of the robot arm (Fig. 2.5). It receives commands from the user via PC (using a TCP /IP connection) or by the manual control unit (MCP) included in the system.

2.2.1. Control Unit

The Control Unit is composed by a central processing unit or CPU. The CPU drives the robot through six power amplifiers, each one dedicated to each axis of the arm. It has a power unit for the electrical energy conversion, in order to provide the required voltage for each moving element. Since all these elements are located within an enclosed and reduced space, it has incorporated a heat exchanger (air-air), to control the temperature of the CPU boards and power rack.

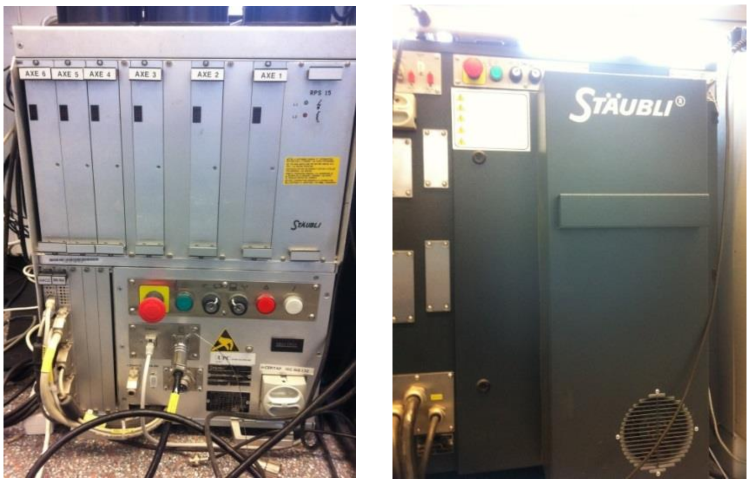


Figure 2.5 - The control units of the robots

2.2.2. MCP

The MCP (Fig. 2.6) is a console that allows communication between the user and the CPU. It consists of a keyboard, LCD screen and the safety button (emergency stop). This button allows blocking all actions in case of an unwanted movement of the robot.

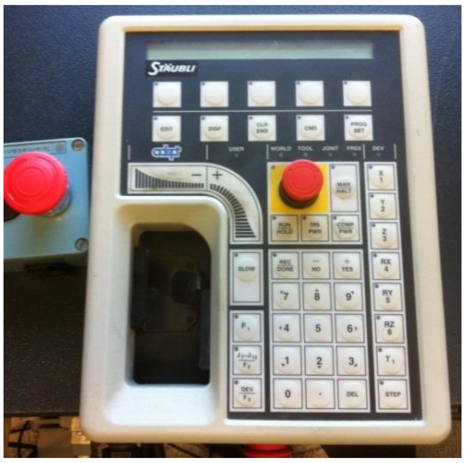


Figure 2.6 - The MCP console that can be used to communicate with the robot

2.4. Digital Video Camera

The camera (Fig. 2.7) used in this project is manufactured by Panasonic, model WV-CP470 of which the technical specifications can be read in the Appendix A.1. It has a resolution of 576 x 768 pixels.

Figure 2.7 - The camera through which the images are acquired

It is important to ensure that the optical properties of the camera remain static, in order to keep its intrinsic parameters constant.

3. THEORETICAL BASIS

In this chapter, basic concepts about camera vision and perspective projection are presented. These are fundamental concepts in order to apply the methods described in next chapters.

First, the meaning and some application areas of camera calibration are explained. The chapter continues with description of the pinhole camera model, on which the entire work is based, and the parameters which have to be identified in order to model the relationship between 3D world and digital image obtained through the camera.

Later on, from literature, some methods for the camera calibration are overviewed and commented.

3.1. Camera calibration

In order to calibrate the position of the robot (or any other object) by vision, the camera calibration is needed. Camera calibration is the process through which a camera is parameterized and, therefore, the information given by its images can be properly treated. The parameters of the camera are classified according to:

- those concerning the internal geometry and the optical properties of the camera (intrinsic parameters)
- those concerning the relative position and orientation of the camera with respect to another reference coordinate system (extrinsic parameters).

Camera calibration is eventually used to achieve different objectives in many applications [1], for example:

- 3D reconstruction: Having a set of images of the same scene, the three-dimensional metric coordinates of the points can be calculated. This is possible by two different ways: multiple images of the same scene can be taken either using multiple cameras in static position, or a single moving camera taking multiple images of the same scene.
- Visual inspection: After obtaining measurements of an object by reconstructing its 3D points, they can be compared to a reference model in order to detect significant parameters of any imperfection or incongruence. This technique is used for example in industry to make quality control of manufactured parts, where can be determined defects like cracks or dents.
- Object localization: By calibrating a camera, it is possible to obtain its position and orientation with respect to the reference frame of another object. Therefore, considering various image points from different objects, the relative position among these objects can be determined.

- Camera localization: If a camera is placed on a robot arm or on a mobile robot to locate some known landmarks in the scene, the position and orientation of the camera can be determined at any moment and, therefore, the position of the robot.

More conceptually, two main operations [2] can be achieved from the calibration process:

- To obtain the 3D world coordinates from the 2D coordinates on the image. The objective can be either finding the position of the objects in the scene or finding the position of a moving camera.
- To infer 2D image coordinates from 3D points. By observing an image, a hypothesis of the state of world can be confirmed or verified. If the image coordinates and the 3D coordinates of the object match, the hypothesis is confirmed. In order to execute this comparison, it is necessary to have the intrinsic and the extrinsic coordinates of the camera.

Vast literature can be found about several techniques to obtain the parameters of a camera, which can be classified according to different criteria [1]:

- Linear vs non-linear techniques. Linear techniques usually do not consider the lens distortion, while the non-linear do.
- Implicit vs explicit techniques. Implicit calibration is the process of calibrating a camera without explicitly computing its physical parameters, contrary to the explicit calibration.
- Points based vs vanishing lines based techniques. The techniques based on points have set of coordinates as computing inputs. The techniques based on vanishing lines do not consider the coordinates of single point, but the vanishing points and the vanishing lines.

3.2. Pinhole model

A pinhole camera [6], also known as "dark chamber", is a simple optical imaging device. It has a shape of a closed box or chamber. In one of its sides there is a small hole which, via the rectilinear propagation of light, creates an image of the outside space in the opposite side of the box.

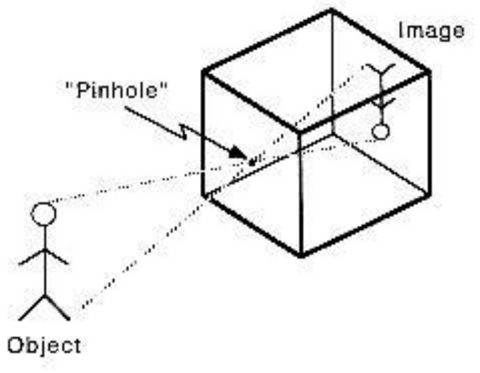


Figure 3.1 - A simple representation of the pinhole camera. All the rays coming from the object pass through a hole and form the reversed image on the back side

The size of projection on the image plane (also called "projection plane") depends on a camera parameter called focal length. This parameter corresponds to the distance between the pinhole plane and the image plane.

Referring to Fig. 3.2, f is the focal length, Z is the distance between the object and the pinhole, X is the object longitude and x is the corresponding longitude on the image plane. The optical axis is the ray orthogonal to the image plane passing through the hole.

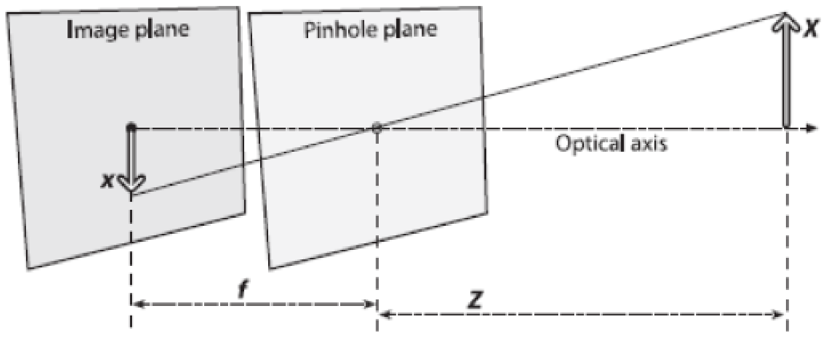


Figura 3.2 - The pinhole camera model with the graphic explanation of the formula (1)

The correspondence between X and its projection x is simply expressed considering the two triangles formed by the optical axis and the ray of projection (1).

$$-x = \frac{fX}{Z} \qquad (1)$$

A variant of this model is considered, as shown in Fig. 3.3. The variant consists of swapping the image plane and the pinhole plane, so that the object is not projected upside down on the image plane. The pinhole has become the center of projection (or optical center), that is the intersection of all the rays from each object point.

This variant is a simplification of the original model shown in Fig. 3.2. It provides the simplest expression of the geometric relationship between a 3D point and its 2D corresponding projection onto the image plane. The coordinates in the 3D world are expressed with reference to the camera reference system.

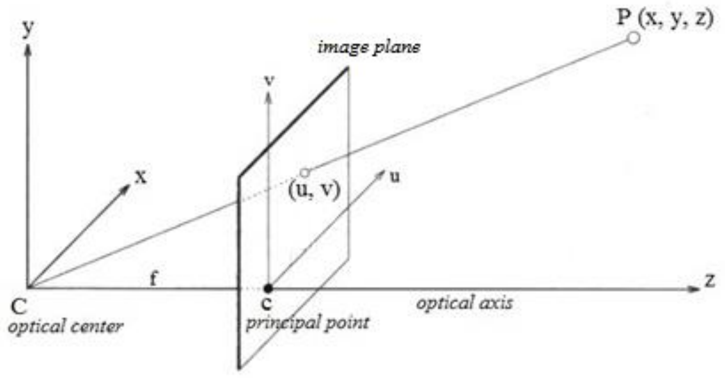


Figure 3.3 - The reviewed pinhole camera model

The ***optical center*** C is the center of the perspective projection. The points belonging to the 3D world are projected onto the ***image plane***, which is parallel to the xy plane of the camera reference frame. The ***optical axis*** is the line perpendicular to the image plane passing through the optical center. The ***principal point*** c is the intersection between optical axis and image plane. The ***focal length*** f is the distance between optical center and principal point. Each 3D point *P(X, Y, Z)* is corresponded by a 2D point *π(u, v)* in the image plane [3].

In this model, it is assumed that the optical axis is collinear to the z-axis and the optical center is located at the origin of the 3D coordinate system.

The projection of a 3D world point $(X, Y, Z)^T$ onto the image plane at pixel position $(u, v)^T$ can be written as

$$u = f\frac{X}{Z} \qquad (2)$$

$$v = f\frac{Y}{Z} \qquad (3)$$

where f denotes the focal length. The previous relation can be reformulated, by adding 1 as fourth coordinate to the point *(X, Y, Z)*^{T}. The coordinates *(X, Y, Z, 1)*^{T} are named homogeneous coordinates.

Then, relations (2) and (3) can be expressed in matricial notation by

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad (4)$$

where λ is the homogeneous scaling factor.

3.3. Intrinsic parameters

The intrinsic parameters are the internal parameters of the camera, which allow linking 3D coordinates in the camera reference frame with the corresponding pixel coordinates of an image point. Intrinsic parameters are described in the following sections.

3.2.1. Focal length

The focal length, in the perfect model described above, is the distance f from the optical center to the image plane, and it is usually noted in pixel units by f_x and f_y.

$$f_x = f s_x, \quad f_y = f s_y \qquad (5)$$

The image scale factors, noted s_x and s_y, depend on the pixel aspect ratio. If the pixels are perfectly squared, f_x and f_y have the same value. Then, the relation between a 3D point and its projection in pixel is:

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \qquad (6)$$

3.2.2. Principal point

The principal point corresponds to the image coordinates on which the optical axis intersects the image plane. It is expressed by pixel coordinates (u_0, v_0).

It was assumed in (6) that the origin of the pixel coordinate system corresponds to the principal point, but the pixel coordinates system have usually its origin at the upper-left corner of the image. Then, the principal point may not have coordinates (0, 0). Integrating the coordinates (u_0, v_0) to the matrix notation (6), the projection equation becomes

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{7}$$

3.2.3. Matrix of intrinsic parameters

The intrinsic parameters described above are included in a matrix that expresses the relationship between a point *P* referred to the camera coordinates system and its projection *(u, v)* in pixel units on the image plane, as seen in Fig. 3.3. This relationship is expressed as:

$$\lambda P_I = IP \tag{8}$$

where λ is the homogeneous scaling factor, P is the 3D point, P_I is the pixel on the image. The matrix *I*, expressed as

$$I = \begin{pmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{9}$$

And is usually named as the intrinsic parameter matrix for the camera.

3.2.4. Lens distortion

The lens distortion [4] is another parameter that alters the correspondence from 3D points and their projections. Two main types of distortion are described: radial distortion and tangential distortion.

The **radial distortion** is a matter of the shape of the lens. Usually, the location of the pixels is distorted near the edges of the lens, so obtaining the effect named "fish-eye".

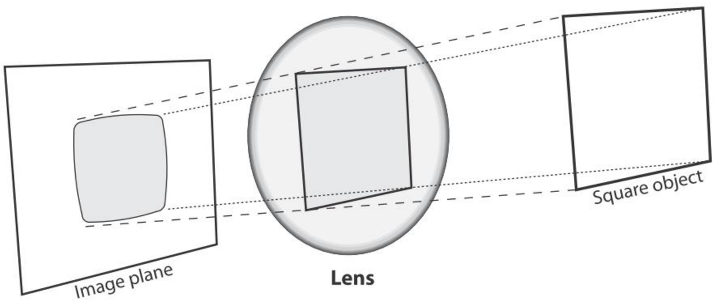


Figure 3.4 - A simple illustration of the radial distortion. It causes curving of the object near the lens edges

Radial distortion is null at the principal point and increases near the edges. This distortion can be characterized by the first few terms of a Taylor series. In general the first two terms of the series are considered, the first of which is conventionally called k_1 and the second k_2. In the case of highly distorted cameras, also a third term, called k_3, is considered.

The following equations usually are considered in order to model the influence of the radial distortion:

$$x_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$\qquad\qquad (10)$$
$$y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

The **tangential distortion** is due to manufacturing defects for which the lens is not fully parallel to the image plane. Tangential distortion is conventionally characterized by two additional parameters, p_1 and p_2, and is so modeled:

$$x_{corrected} = x + [2p_1 y + p_2(r^2 + 2x^2)]$$
$$\qquad\qquad (11)$$
$$y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 x]$$

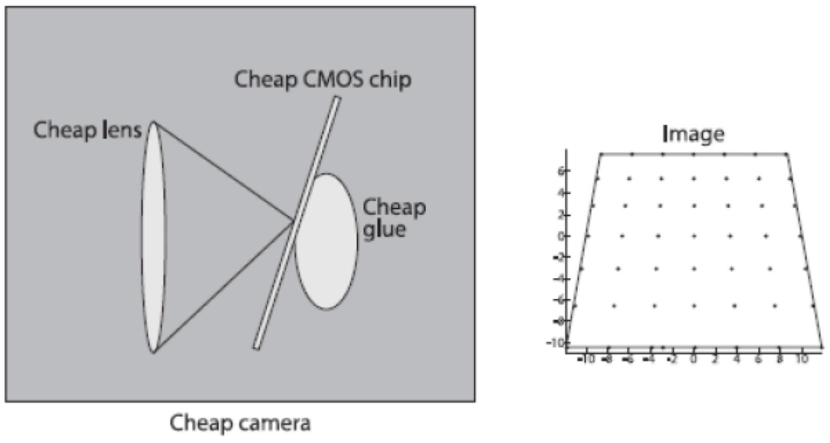


Figure 3.5 - Effects of the tangential distortion, caused by imperfect parallelism between lens and image plane

There are usually five distortion coefficients considered in the calibration techniques (see Camera Calibration Tool in 4.3 and OpenCV in 4.4). They are typically bundled into one distortion vector, that is a 5-by-1 matrix containing, k_1, k_2, k_3, p_1 and p_2.

3.4. Extrinsic parameters

The extrinsic parameters define the location and the orientation of the camera reference frame with respect to a known world reference frame. This relationship is defined by a rotation matrix and a translation vector.

3.4.1. Rotation matrix

The rotation of the camera axes with respect to a reference coordinate system is expressed by a 3-by-3 matrix composed as follows:

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} cos\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{pmatrix} \begin{pmatrix} cos\omega & sin\omega & 0 \\ -sin\omega & cos\omega & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (12)$$

The values θ, ϕ, ω are the rotation angles of the camera, related respectively to its x, y, and z axes.

The rotation matrix has two important properties:

- the dot product of any pair of rows or any pair of columns is 0;
- the norm of each row and each column is 1 (each row or column is an unit vector)

These properties are not casual. In fact, the columns of the rotation matrices are the unit vectors $\hat{\imath}$, $\hat{\jmath}$ and $\hat{k}$.

$$R = (\hat{\imath} \quad \hat{\jmath} \quad \hat{k}) \quad (13)$$

These vectors correspond to the directions of the axes of the camera expressed in the robot's coordinate frame.

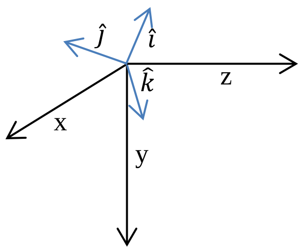


Figure 3.6 - The effects of a rotation: the x, y, z axis compose the camera reference frame. The unit vectors i, j, k are the direction of the axes after the rotation.

3.4.2. Translation vector

The translation vector is a 3-by-1 matrix composed as follows:

$$T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \quad (14)$$

The values t_x, t_y and t_z correspond to the center of the coordinate system of the camera expressed in the robot's reference frame.

3.4.3. Transformation matrix

In order to express a change of coordinate system (from robot to camera and vice versa) a 4-by-4 matrix containing the extrinsic camera parameters is considered (15).

$$E = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (15)$$

This matrix is used to convert a 3D point P_A belonging to a reference system (for example the robot A), into the corresponding coordinates P_B of another reference system (for example the robot B) (16). The points must be expressed in homogeneous coordinates.

$$P_B = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} * P_A \qquad (16)$$

3.5. Relating 3D points and camera pixels

In this section is explained the complete procedure with which the coordinates of a 3D point in the world reference frame are linked to the coordinates (u, v) of its projection onto the image plane. In order to express this correspondence on which the calibration methods are based, four steps are usually defined (see Fig. 3.4):

Step 1. It consists of modeling the correspondence, as translation and rotation, between the camera coordinate system and the world coordinate system. Then, each point $P_W = (X_W, Y_W, Z_W)$ of the world coordinate system is related to the camera coordinate system, obtaining $P_C = (X_C, Y_C, Z_C)$.

Step 2. It consists of modeling the projection of the point (X_C, Y_C, Z_C). on the image plane in order to obtain the 2D projection P_u, undistorted.

Step 3. It expresses the influence of the lens distortion, introducing a disparity with the real projection. Then the point P_u is transformed into the real projection P_d, which is P_u distorted.

Step 4. It consists of carrying out another coordinate system transformation in order to change from the metric coordinate system of the camera to the image coordinate system of the computer in pixels, obtaining the final coordinates (u, v).

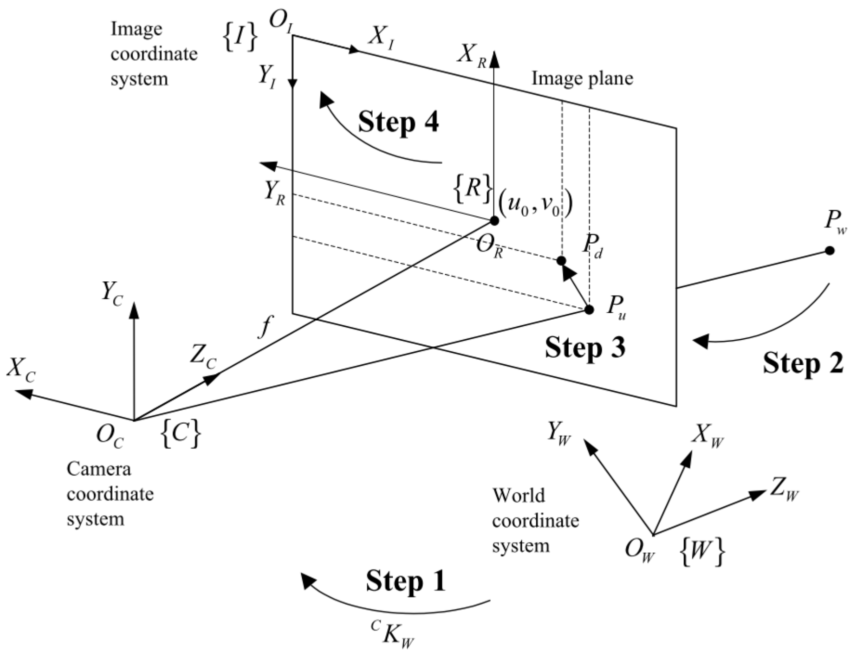


Figure 3.4 - Graphical representation of the four steps to calibrate a camera

Given the intrinsic and the extrinsic camera parameters in matricial form (written as I and E) and the 3D point, the relationship between a 3D point and its 2D projection is expressed, without considering the distortion, as

$$\lambda P_c = \begin{pmatrix} \lambda u \\ \lambda \text{v} \\ \lambda \end{pmatrix} = IEP = IE \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{17}$$

then

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 & 0 \\ 0 & f_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{18}$$

3.6. Calibration techniques

In this section, some techniques found in literature are described and discussed. The implementation of an alternative method has required a previous study of the principal existing methods of camera calibration. Each one is overviewed an commented stressing the reasons why its procedures is not compatible with the purposes of the alternative method (Section 1.2).

The methods of Hall and Faugeras-Toscani without distortion do not consider the lens distortion in the model definition. In order to obtain the parameters of the model, they solve a linear system using a least squares technique.

The method of Faugeras-Toscani with distortion and Tsai are two non-linear methods. The reason is that they consider some distortion coefficients in the model definition. These methods, in order to obtain the parameters, consist of two stages. In the first stage they obtain an initial estimation of the parameters by linear approximation. In the second stage, an iterative method is used in order to optimize the parameters.

3.7.3. The method of Hall

The method of Hall solves a simple linear system by using the least squares method. It considers the following expression to relate a 3D point (X, Y, Z) to its projection (u, v):

$$\begin{pmatrix} \lambda u \\ \lambda v \\ \lambda \end{pmatrix} = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \tag{19}$$

The matrix

$$P = \begin{pmatrix} P_{11} & P_{12} & P_{13} & P_{14} \\ P_{21} & P_{22} & P_{23} & P_{24} \\ P_{31} & P_{32} & P_{33} & P_{34} \end{pmatrix} \tag{20}$$

is the projection matrix that allows calculating the projection on the image plane for each 3D point in the world reference frame.

By expressing the relation in the form

$$u = \frac{P_{11}X + P_{12}Y + P_{13}Z + P_{14}}{P_{31}X + P_{32}Y + P_{33}Z + P_{34}} \tag{21}$$

$$v = \frac{P_{21}X + P_{22}Y + P_{23}Z + P_{24}}{P_{31}X + P_{32}Y + P_{33}Z + P_{34}} \tag{22}$$

for each of the N points, a linear system with 2N equations and 12 unknowns is created and solved obtaining the values of the P matrix.

The projection matrix A can be decomposed in order to obtain the intrinsic matrix I and the extrinsic matrix E,

$$P = IE = \begin{pmatrix} f_x & \tau & o_x \\ 0 & f_y & o_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix} \tag{23}$$

One of the constraints of the alternative method is the avoidance of computing the intrinsic parameters. The method of Hall forces to extract them from the projection matrix; hence it is does not verify the constrains fixed for the alternative method to be developed.

3.7.4. The method of Faugeras-Toscani without distortion

This method allows obtaining the extrinsic and the intrinsic parameters by solving a linear system. It defines the model as

$$\begin{pmatrix} \lambda u \\ \lambda \mathrm{v} \\ \lambda \end{pmatrix} = IE \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{24}$$

$$\lambda \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & o_x & 0 \\ 0 & f_y & o_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \tag{25}$$

The matrices I and E are multiplied in order to obtain another matrix:

$$A = \begin{pmatrix} f_x r_1 + o_x r_3 & f_x t_x + o_x t_z \\ f_y r_2 + o_y r_3 & f_y t_y + o_y t_z \\ r_3 & t_z \end{pmatrix} = \begin{pmatrix} A_1 & A_{14} \\ A_2 & A_{24} \\ A_3 & A_{34} \end{pmatrix} \tag{26}$$

Then, a linear system is solved by using a least squares method and obtain the values of the A matrix, from which the single parameters are extracted one by one.

Then, it is impossible to obtain the extrinsic parameters without first obtaining the intrinsic parameters. For this reason, the implementation of this method has been rejected.

3.7.5. The method of Faugeras-Toscani with distortion

The linear method of Faugeras can be made with better accuracy considering the distortion. The fact of introducing the distortion in the model forces to change the solving technique. The linear least squares technique is replaced by an iterative method because the system becomes non-linear.

In order to apply this method, an initial estimation is necessary. It is obtained by applying the Faugeras-Toscani method without distortion. The iterative method with this initial estimation is executed in order to minimize the error, which is the difference between the initial parameters and the computed ones during iterations.

The distortion is an intrinsic parameter, then it cannot be included in the geometrical calculus of the alternative method. Then, the method of Faugeras-Toscani with distortion gives no contribution for the purposes of this project.

3.7.6. The method of Tsai

This method models the radial distortion, but assumes that some parameters of the camera are known, since they are provided by manufacturers.

The method can be divided into two stages. In the first stage all extrinsic parameters except for t_z are computed by using a constraint. The constraint is a condition of parallelism. In the second stage all missing parameters are evaluated by non-linear optimization.

Also this method has some feature not compatible with the idea for an alternative method: some parameters are assumed to be known in order to compute, by solving the initial linear system. For the alternative method it is avoided any consideration of the internal parameters of the camera.

In conclusion, none of the methods just examined can provide an effective aid to formulate an alternative method that avoid the computation of the intrinsic parameters. Thus, the objectives described in Section 1.2 have been pursued by formulating a type of reasoning completely different from those proposed in this chapter, as described in Chapter 6.

4. REFERENCE METHOD FOR ROBOT CALIBRATION

In order to validate the results of the calibration procedure proposed in this thesis, it is compared to an existing methodology. The main reason is to evaluate the level of accuracy obtained, determining whether the differences are due to the analytical assumptions and their implementation (the equipment and devices used are the same in both cases, hence deviations in results are only due to computational techniques). Therefore, this method working as a reference also needs to be implemented, as described in this chapter.

4.1. Procedure description

The calibration procedure chosen as the "reference method" consists of computing position and orientation of the camera with respect to the robots, in order to obtain the position of one robot with respect to the other. The reference method requires computing the internal features of the camera in order to be located with respect to the robots.

Therefore, procedure is split into three stages: intrinsic camera calibration, extrinsic camera calibration with respect to each robot and finally the calibration of the position of the robots.

The intrinsic and extrinsic parameters of the camera are obtained separately by means of different software applications. The software tool used to achieve the extrinsic calibration requires the results of the intrinsic calibration as input.

In the **first stage** the intrinsic calibration is made by means of a MATLAB tool called "Camera Calibration Toolbox". The process consists of taking several images of a chessboard on which a set of points is detected in order to compute the intrinsic parameters of the camera, as described in 4.3. The resulting intrinsic parameters are manually stored in text files and they are inputs in the second stage.

In the **second stage,** the extrinsic calibration is achieved by means of an OpenCV function, the inputs of which are

- The 3D coordinates of the points on which the robot arms have been moved,
- The image projections of the points cited above, obtained through the Visual Studio interface described in Chapter 6,
- The intrinsic parameters of the camera, obtained at first stage.

In the **third stage**, the position of the robots, one with respect to the other, is obtained. Given the transformation matrices from the camera with respect to each robot, it is only necessary to apply a trivial formula to obtain the position of one robot with respect to the other (Fig. 4.6).

By means of Visual Studio, two interfaces have been developed for image processing and data treatment necessary to implement the reference method. The first interface is in charge of capturing the images of the infrared LED attached to the robots end effector, so that its coordinates (X, Y, Z) in the robots reference frame and its projection (u, v) in the image frame are saved in a text file for each position. The format of the output file fits the input for the second interface, which will compute the extrinsic calibration of the camera with respect to each robot.

4.2. Employed tools

As previously announced, MATLAB and Visual Studio C++ (including OpenCV libraries) are the platforms chosen to implement the reference method.

MATLAB. A specific MATLAB tool called "Camera Calibration Toolbox", as explained in 4.3, allows the computation of the intrinsic parameters by manually setting relevant points of images (25 in our case) of a chessboard with known dimensions.

Visual Studio. This environment and C++ language are designed for an object-oriented programming. It allows simplifying the interface design and the implementation of its functions. The Visual Studio environment offers many predefined tools that ease the work (dialog boxes, text controls, picture controls, etc…), as well as specific libraries.

OpenCv (Open Source Computer Vision)**.** It consists of a set of libraries for computer vision. These libraries include several functions helpful for the image processing and for camera calibration.

In the following, the whole procedure by using each single tool is explained.

4.3. Intrinsic parameters with Camera Calibration Toolbox

The Camera Calibration Toolbox [5], developed for MATLAB, is a very useful tool for calibration. This tool works on several images of a planar calibration object, in this case a 4-by-5 chessboard. It allows marking a set of known points on each image in order to find the correspondences between the position of 3D points and their projections on the image.

However, the Camera Calibration Toolbox has a practical disadvantage: the procedure is not very user friendly, because each image requires accurate manual operations.

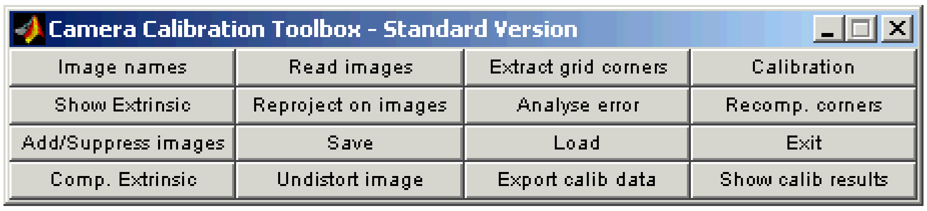


Figure 4.1 - Main interface of the Camera Calibration Tool

As preliminary step, a certain number of images of the chessboard (in this case, 25) are taken, using the camera to be calibrated. In each image, the chessboard is captured in a different position and orientation, in order to better evaluate the effects of the distortion, hence improving the accuracy of the parameters obtained.

A function called "Extract grid corners" is used to mark the points on the chessboard that has a structured pattern with known dimensions. The program estimates the positions of vertices of the internal squares from a few points marked by the user, considering no distortion. Then, by manually correcting a distortion parameter, the user can correct the positions of the vertices.

Firstly, the application asks the user to enter the number of squares of the chessboard along the {x,y} axis and their size (x,y) in millimeters. In this case, the size is 30x30. Obviously, another input is the image of the chessboard.

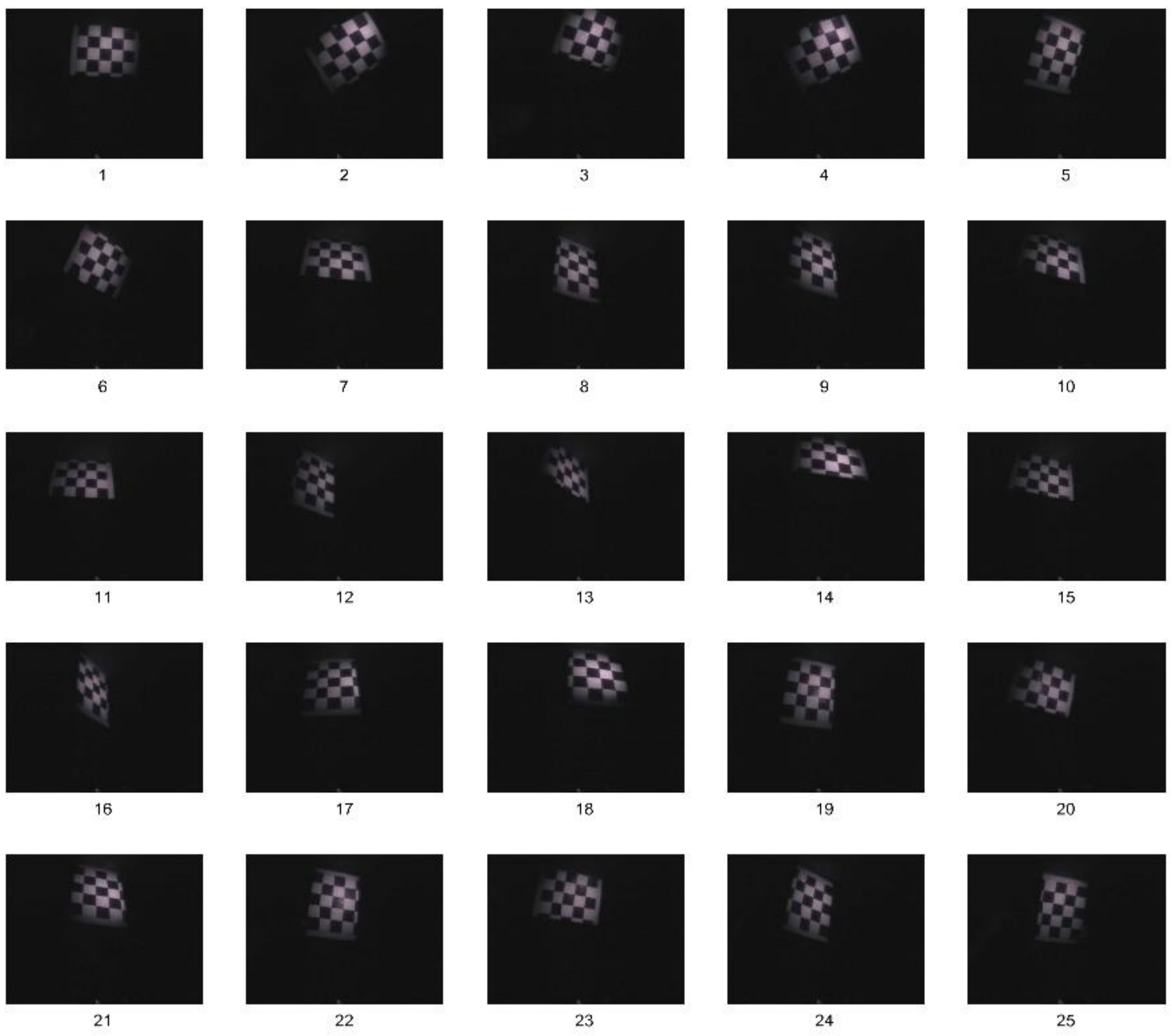


Figure 4.2 - The set of 25 images of the chessboard. These images have been taken using an IR illumination, since the camera has been calibrated with an IR filter.

Then, the user has to click the four corners on the chessboard to define the world coordinate system. The origin of the world coordinate system is considered as one of these corners. The user can choose the corner to be set as origin, but it must be the same for all images.

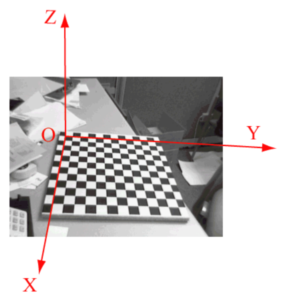


Figure 4.3 – A sample image with the axis of the world coordinate system

After fixing the axis of the world reference frame, the tool shows the user the image with the predicted corners. The red crosses positioned initially on the image correspond to the corners that would result with no distortion.

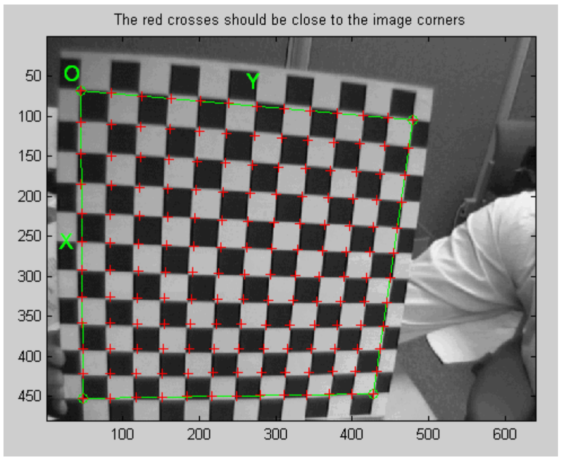


Figure 4.4 – A sample image with the corners initially placed by the tool, not close enough to the real corners due to distortion

If the estimated corners are not close enough to the real image corners, it is necessary to "help" the software to refine the predicted corners by making a better guess of the corner locations. The user has to manually enter an initial value for the first order coefficient of the radial distortion kc (that is the first entry of the 5-by-1 distortion coefficient vector). Once the user updates the coefficient value, the chessboard is shown again with updated red crosses. The user can repeat this process as many times as wanted, until the estimation is satisfactory.

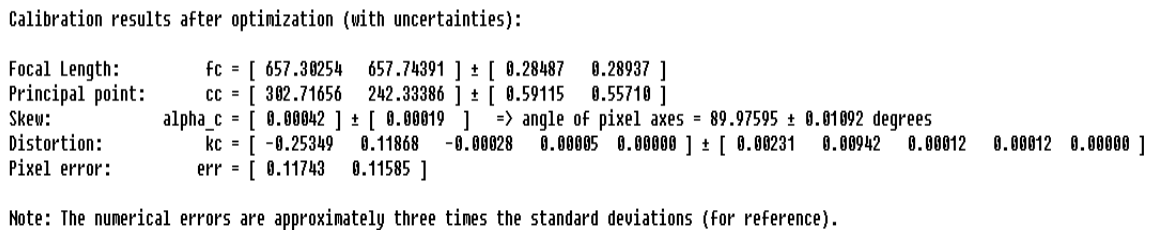
If lens distortions are really too severe, the simple guiding tool based on a single distortion coefficient kc may not be sufficient to provide good enough guesses for the corner locations. For those few difficult cases, the toolbox includes a script that allows a completely manual corner extraction (one click per corner). For this project, however, there was no need to resort to this solution.

For each image identified by i, two arrays are obtained, in the following forms:

$$P_i = \begin{bmatrix} X_i(1) & \cdots & X_i(j) & \cdots & X_i(M_i) \\ Y_i(1) & \cdots & Y_i(j) & \cdots & Y_i(M_i) \\ Z_i(1) & \cdots & Z_i(j) & \cdots & Z_i(M_i) \end{bmatrix} \qquad (26)$$

$$\pi_i = \begin{bmatrix} x_i(1) & \cdots & x_i(j) & \cdots & x_i(M_i) \\ y_i(1) & \cdots & y_i(j) & \cdots & y_i(M_i) \end{bmatrix} \qquad (27)$$

In P_i the coordinates $(X_i(j), Y_i(j), Z_i(j))$ of any 3D point corresponding to the chessboard's corners are stored. In π_i, the corresponding projections in pixel $(x_i(j), y_i(j))$ are stored. These arrays are inputs for the automatic intrinsic calibration procedure. An example of the final output is the following:



```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 657.30254   657.74391 ] ± [ 0.28487   0.28937 ]
Principal point:       cc = [ 302.71656   242.33386 ] ± [ 0.59115   0.55710 ]
Skew:             alpha_c = [ 0.00042 ] ± [ 0.00019  ]   => angle of pixel axes = 89.97595 ± 0.01092 degrees
Distortion:            kc = [ -0.25349   0.11868   -0.00028   0.00005  0.00000 ] ± [ 0.00231   0.00942   0.00012   0.00012  0.00000 ]
Pixel error:          err = [ 0.11743   0.11585 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

The parameters of interest are the intrinsic parameters described in 3.3. In order to evaluate the goodness of the results obtained, the “Pixel error” is considered. In the case above, the error is less than 0.5 pixels in either horizontal or vertical plane, indicating that the distortion corrections have been made with enough accuracy.

4.4. Extrinsic calibration with OpenCV

This section describes the design and implementation of the interface that allows obtaining the camera extrinsic calibration by means of the OpenCV libraries.

This interface calls an OpenCV function that uses the correspondences between 3D positions referred to a coordinate system and their image projections to determine the position and orientation of the camera with respect to that coordinate system. In order to achieve the extrinsic calibration, the function also needs the intrinsic parameters of the camera.

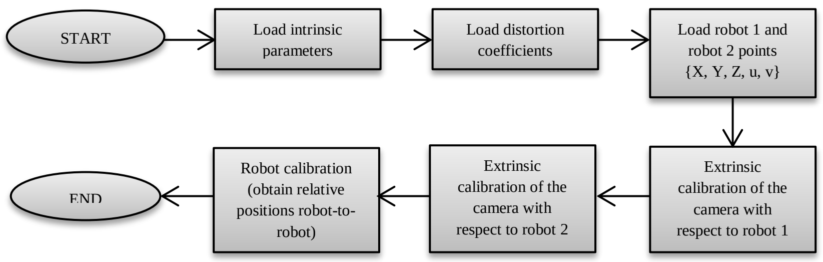

START
Load intrinsic parameters
Load distortion coefficients
Load robot 1 and robot 2 points {X, Y, Z, u, v}
Extrinsic calibration of the camera with respect to robot 1
Extrinsic calibration of the camera with respect to robot 2
Robot calibration (obtain relative positions robot-to-robot)
END

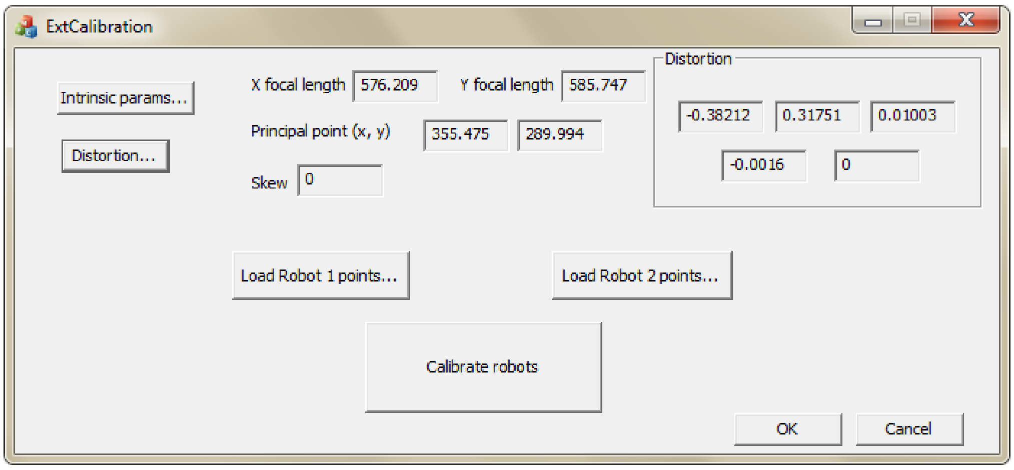
Figure 4.5 - Flow diagram of the extrinsic calibration in the reference method

From two files called “intrinsic.txt” and “distortion.txt”, the results given by the Camera Calibration Toolbox are loaded, while from other two files called “robotA.txt” and “robotB.txt”, the points acquired for both robots are taken by means of the interface described in Chapter 6.

The functions used for the calibration process (matricial operations, calibration and matrix conversion) belong to the OpenCV libraries.

The interface, shown in Fig. 4.7, is composed by:

1. a button for loading the intrinsic parameters. These parameters are taken from a text file (that contains a single row {x focal length, y focal length, x principal point, y principal point, skew}. These values are put on a CvMat object, which corresponds to the intrinsic parameters matrix.
2. a button for loading the 1-by-5 vector of the distortion coefficients. These values are taken from a text file and stored in a CvMat object.
3. two buttons for loading the points {X, Y, Z, u, v} taken for each robot. The points are separated between the world points (X, Y, Z) and the projections (u, v). The N world points are stored in a N-by-3 matrix, while the N projections are stored in a N-by-2 matrix. This is done for both robots.
4. the button "Calibrate robots", that starts the computation in order to give in output the extrinsic parameters from camera to both robots, then the transformation matrix that relates the camera to the robots, finally the transformation matrix from one robot to the other.


ExtCalibration

X focal length 576.209

Y focal length 585.747

Principal point (x, y) 355.475 289.994

Skew 0

Distortion

-0.38212 0.31751 0.01003

-0.0016 0

Figure 4.7 - User interface for the extrinsic calibration

The extrinsic calibration from the camera with respect to one robot is achieved by calling an OpenCV function called *cvFindExtrinsicCameraParams2()*. This function has the following prototype:

```
void cvFindExtrinsicCameraParams2(

        const CvMat*    object_points,
        const CvMat*    image_points,
        const CvMat*    intrinsic_matrix,
        const CvMat*    distortion_coeffs,
        CvMat*          rotation_vector,
        CvMat*          translation_vector

);
```

The function *cvFindExtrinsicCameraParams2()* takes in input the N-by-3 matrix *object_points*, that contains the (X, Y, Z) points given by the robot, the N-by-2 matrix *image_points* that contains the projections (u, v), the 3-by-3 matrix of the intrinsic parameters, the 1-by-5 vector of the distortion coefficients.

The output results of this function are the extrinsic parameters, split into the vectors *rotation_vector* and *translation_vector*. Although the rotation vector is given by a 1-by-3 matrix, it can be converted into the 3-by-3 rotation matrix by means of the *cvRodrigues2()* function. The translation vector contains the offsets along each axis from the optical center to the origin of the robot reference frame.

The extrinsic parameters are stored in two matrices 4-by-4 (transformation matrices), one for each robot, as described in 3.4.3.

4.5. Robot calibration

Once the extrinsic calibration of the camera is made with respect to both robots, the relative position from the robot A with respect to the robot B (or vice versa) is obtained as another 4-by-4 transformation matrix.

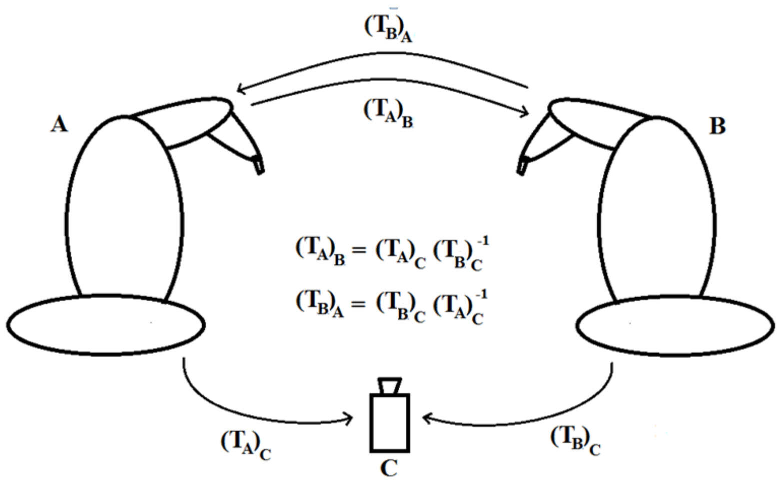

$(T_B)_A$
$(T_A)_B$
A
B
$(T_A)_B = (T_A)_C\,(T_B)_C^{-1}$
$(T_B)_A = (T_B)_C\,(T_A)_C^{-1}$
$(T_A)_C$
$(T_B)_C$
C

Figure 4.6 - Homogeneous transformations among robots and camera

$(T_C)_A$ and $(T_C)_B$ are the transformation matrices from camera C with respect to robot A and B, respectively. According to Fig 4.6, the transformation from the robot A with respect to the robot B can be obtained as follows:

$$(T_A)_B = (T_A)_C * (T_B)_C^{-1} = (T_C)_A^{-1} * (T_C)_B \tag{28}$$

5. ALTERNATIVE METHOD FOR ROBOT CALIBRATION

In this chapter, an alternative method to calibrate the positions of the robots with respect to the camera is presented and explained.

As reported in 3.6, in order to design the alternative method, some existing techniques have been evaluated. Generally, the methods found in literature that operate by vision involve the computation of the internal characteristics of the camera.

The alternative method is designed in order to directly obtain the extrinsic calibration of the camera without considering in any way its intrinsic parameters, because it is expected that they are not necessary to calibrate the position of the robots.

This method is based on geometry. The 3D points to be captured by the camera are arranged in a particular configuration and verify a set of conditions necessary to solve the system. For this reason, they cannot be randomly taken, as it is stated below. The procedure consists of three principal steps:

1. Computing the translation: the optical center C_P of the camera is computed, with respect to each robot's coordinate system, as intersection of four straight lines (Fig. 5.3, 5.4) passing through a chosen set of points (Section 5.1). The solution obtained is the translation vector *t* from the robot with respect to the camera;
2. Computing the orientation: the directions of the camera axes are computed, with respect to each robot's reference frame. In order to find the axes, the angles measured in the projected polygon are considered (Fig. 5.5, 5.8) and some conditions are imposed by solving non-linear systems (Section 5.2). These directions are normalized and form the rotation matrix R from the robot, with respect to the camera;
3. The transformation matrix from one robot with respect to the other is computed (Section 5.3). This is obtained by applying the final formula (28).

In order to acquire the (X, Y, Z) coordinates of the points and their corresponding projections (u, v) on the image plane, the first interface developed by means of Visual Studio (Chapter 6) has been used.

In next paragraphs, the method is explained in detail.

5.1. Finding the optical center

The first step consists of finding the optical center of the camera, which indeed is the origin of the camera coordinate system. The optical center is obtained as intersection of a set of four straight lines passing through multiple 3D points with the same (u, v) projection.

In order to find the four straight lines which intersect on optical center, the points have been chosen using two different configurations. In the initial configuration (Fig. 5.3a), eight points have been considered. They are located in two parallel planes, so that the four points for each plane form a square. The square shape has not been chosen for technical reasons, but for its simplicity.

The initial configuration was designed also to find the camera orientation. The reason why a second configuration (Section 5.2.2) has been designed is to improve the accuracy in computing the optical center by increasing the number of points (constraints) through which the lines have to pass.

The second configuration (Fig. 5.4) consists of four sets of points with the same projections (u, v), without needing to belong to predefined planes. This last configuration allows finding the center of projection more accurately.

5.1.1. First configuration

For the initial configuration, the four straight lines are obtained considering four pairs of points, the projection (u, v) of which coincide on the image. The points are placed in two different planes and these planes are parallel. Each plane contains four points arranged as the vertices of a square.

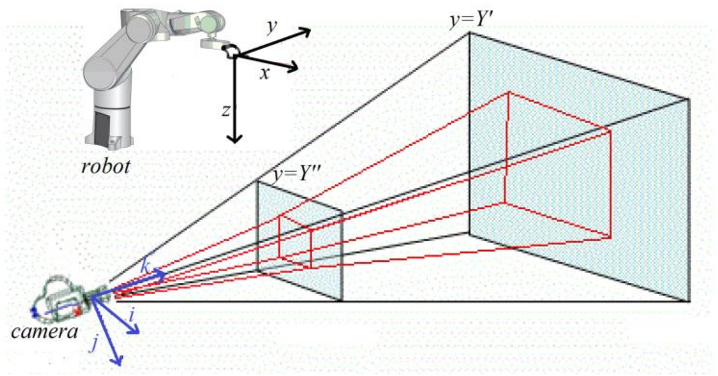


Figure 5.1 - First configuration of the 3D points of study. The lines passing through the corresponding vertices of the squares have the same projection; hence they intersect in the optical center.

Each plane on which the points are chosen has a y constant coordinate, with arbitrary values Y' and Y'' (Fig. 5.1).

Through the interface described in Chapter 6, an initial set of four points (Fig. 5.2) is chosen. It can be written in the following notation:

$$\{P_1, P_2, P_3, P_4\} = \{(X_1, Y', Z_1), (X_2, Y', Z_2), (X_3, Y', Z_3), (X_4, Y', Z_4)\} \qquad (29)$$

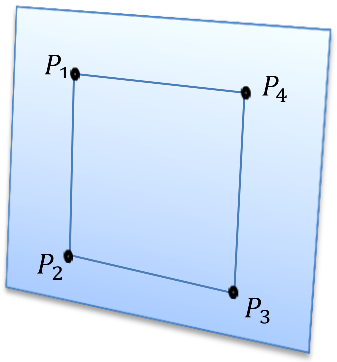


Figure 5.2 – The points P_1, P_2, P_3, P_4 on first plane

For each of these points, the corresponding projection in pixel is saved. Then, at the end of this stage, four correspondences between the 3D points P_i and their projections π_i on the image plane are obtained.

$$\begin{aligned} P_1 &= (X_1, Y', Z_1) \to \pi_1 = (u_1, v_1) \\ P_2 &= (X_2, Y', Z_2) \to \pi_2 = (u_2, v_2) \\ P_3 &= (X_3, Y', Z_3) \to \pi_3 = (u_3, v_3) \\ P_4 &= (X_4, Y', Z_4) \to \pi_4 = (u_4, v_4) \end{aligned} \qquad (29)$$

Another plane parallel to the previous is considered, on which another set of four points is taken. The y coordinate of this new plane is Y'', different from Y'. This second set can be formulated as follows:

$$\{P_5, P_6, P_7, P_8\} = \{(X_5, Y'', Z_5), (X_6, Y'', Z_6), (X_7, Y'', Z_7), (X_8, Y'', Z_8)\} \qquad (30)$$

The *x* and *z* coordinates of these points are chosen so that their projections onto the image plane are the same of those obtained for the first plane:

$$P_5 = (X_5, Y'', Z_5) \to \pi_1 = (u_1, v_1)$$

$$P_6 = (X_6, Y'', Z_6) \to \pi_2 = (u_2, v_2)$$

$$P_7 = (X_7, Y'', Z_7) \rightarrow \pi_3 = (u_3, v_3)$$

$$P_8 = (X_8, Y'', Z_8) \rightarrow \pi_4 = (u_4, v_4) \tag{31}$$

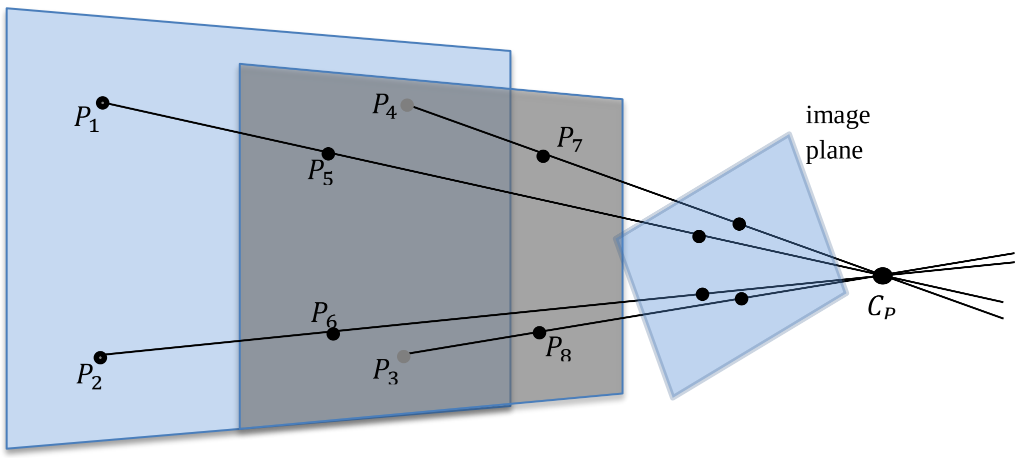


Figure 5.3a - The first configuration of the points and their projections in the image plane

Theoretically, if two points $P_i = (X_i, Y_i, Z_i)$ and $P_j = (X_j, Y_j, Z_j)$ have the same projection $\pi = (u, v)$ on the image plane, it means that P_i and P_j belong to the same ray passing through the camera optical center C_P.

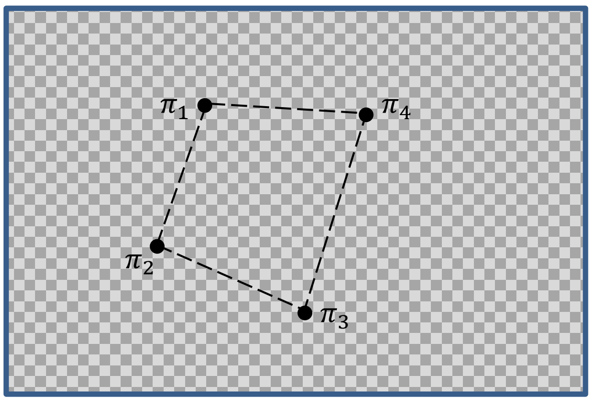


Figure 5.3b - The image resulting from the sample projection shown in figure 5.3a. The projections π_i are expressed in pixel.

The center of projection coincides with the intersection of the four lines passing through the pairs of points. The obtained projections are discretized in pixels, reason why an error is inherently associated to the image resolution of the camera.

Certainly, the four straight lines calculated do not really intersect at a single point due to optical imperfections of real devices. However, it is possible to find a point with minimal distance to the four lines, computing by means least squares.

5.1.2. Improving the computation of the projection center

As explained in 5.1, computing the optical center using the first configuration of points was not stable enough. Two points per line were not sufficient to reliably approximate their real direction, since any tolerance in taking one point can change it. The second configuration designed is expected to minimize this deviation since the constraints for each line become more strong (each line passes through four points instead of only two points).

As shown in Fig 5.4, each line is now defined by four points with the same projection on the image plane. These points neither need to belong to pre-defined planes, nor be arranged as square vertices. However, this new configuration is only used for computing the center of projection.

For each line, the procedure consists of the operations listed below:

1. the robot arm moves to an arbitrary point (X, Y, Z),
2. the (u, v) projections on the image are computed
3. the robot arm moves to other three points far enough from each other, keeping the same projection (u, v)

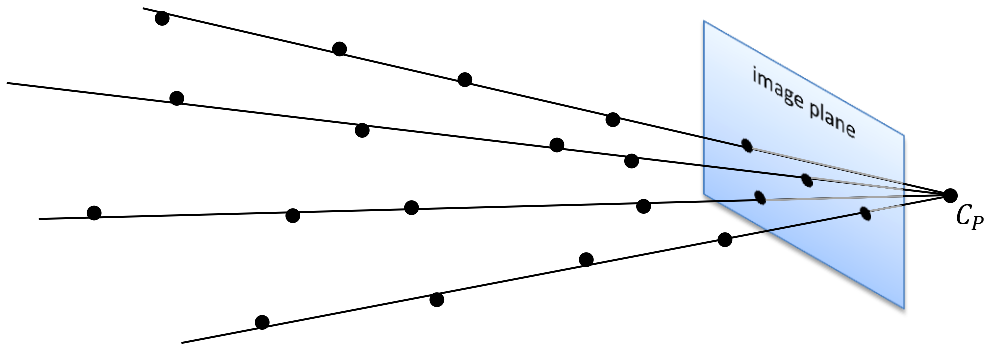


Figure 5.4 - The second configuration, with sixteen 3D points distributed on four lines. The lines are obtained by using a least squares method.

Each set of points with the same projection are not perfectly collinear, because, as said above, there are errors inherent to the system (i.e. the image resolution). In order to find the equations of each line, it is enough to implement a least squares regression (the best fit line), which minimizes the squared distances between the line and the four points having the same projection.

The point of minimal distance among the four lines, is also found by least square distances and considered as the intersection of the four lines.

The intersection $C_P = (X_c, Y_c, Z_c)$ obtained by applying the described method is the optical center referred to the robot frame. Then, the translation vector of the transformation matrix from the robot A (or B) with respect to the camera C is:

$$(t_A)_C = \begin{pmatrix} X_c \\ Y_c \\ Z_c \end{pmatrix} \qquad (32)$$

5.2. Finding the camera orientation

The second step of this method consists of computing the directions of the camera frames with respect to each robot's coordinate system. These directions will be used to obtain the rotation matrix from robot to camera.

In order to find the camera orientation, the points that belong to a single plane of the first configuration are enough, so they are four vertices of a square.

The camera axes are found observing and measuring some properties of the projected polygon. They are found in a specific order:

- the z-axis, that is the normal of the image plane. It is found by solving a non-linear system, in which it is imposed that the projections of the edges of the square on the image plane form the same angles α_i (Fig. 5.5) observed on the image provided by the camera;
- the x-axis, obtained by measuring the angle formed between the bottom edge of the image and an edge of the projected polygon (Fig. 5.8);
- the y-axis, obtained in a simple way from z-axis and x-axis, using the cross product.

The following sections describe the process in detail.

5.2.1. Finding the z-axis

The first axis to be obtained is the z-axis, which corresponds to the normal of the image plane.

The angles of the square defined by the four projections on the image plane are computed (Fig. 5.5). The (u, v) coordinates of the four vertices are known, then the 2D vectors $\overrightarrow{\pi_i \pi_j}$ are obtained in the following simple way:

$$\overrightarrow{\pi_i \pi_j} = \overrightarrow{\pi_j} - \overrightarrow{\pi_i} = (u_j - u_i, v_j - v_i), \qquad i, j \in \{1, \dots, 4\}, \quad i \neq j \qquad (33)$$

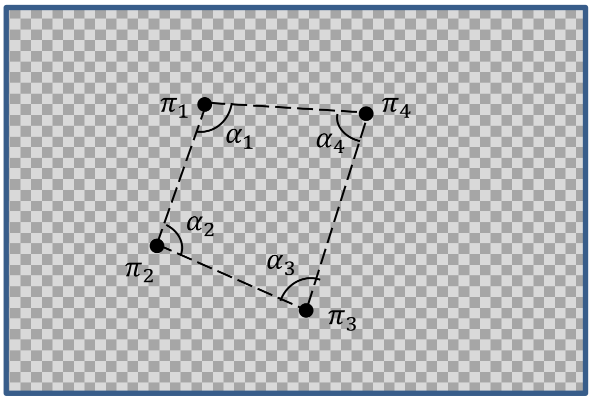


Figure 5.5 - A sample image that shows the angles α_i formed by the edges in the image

The cosines of the four angles α_i are computed, by using the formulas

$$\begin{aligned}
\cos\alpha_1 &= \frac{\overrightarrow{\pi_1\pi_2}\cdot\overrightarrow{\pi_1\pi_4}}{\|\overrightarrow{\pi_1\pi_4}\|\|\overrightarrow{\pi_1\pi_4}\|}\\
\cos\alpha_2 &= \frac{\overrightarrow{\pi_2\pi_3}\cdot\overrightarrow{\pi_2\pi_1}}{\|\overrightarrow{\pi_2\pi_3}\|\|\overrightarrow{\pi_2\pi_1}\|}\\
\cos\alpha_3 &= \frac{\overrightarrow{\pi_3\pi_4}\cdot\overrightarrow{\pi_3\pi_2}}{\|\overrightarrow{\pi_3\pi_4}\|\|\overrightarrow{\pi_3\pi_2}\|}\\
\cos\alpha_4 &= \frac{\overrightarrow{\pi_4\pi_1}\cdot\overrightarrow{\pi_4\pi_3}}{\|\overrightarrow{\pi_4\pi_1}\|\|\overrightarrow{\pi_4\pi_3}\|}
\end{aligned} \tag{34}$$

Considering, as shown in Fig. 5.6, three 3D points P_1, P_2, P_3 belonging to the square, then forming an angle of 90°:

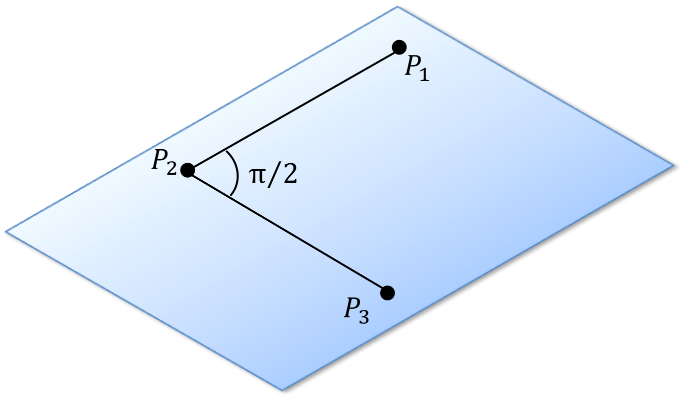


Figure 5.6 - Angle formed by two edges of the square belonging to the plane

These three points belong to a plane φ, the coordinates 'y' of which are constant (this constraint is easily achieved through the control of the robots). The idea is to express the relation that exists between

- the angles formed by the 3D edges of the square $P_1P_2P_3P_4$
- the angles formed by the 3D vectors that are obtained from the projections of those edges onto the image plane.

Then, the objective is to find the normal $\vec{k}$ of the image plane. The image plane to be found is the one complying with the following condition: given the four 3D points that form a square, for each pair of adjoining edges of this square, the angle formed by the projections of this edges on the image plane is equal to the corresponding angle α_i previously computed. Given the normal $\vec{k}$ of a plane, the projection $\overrightarrow{\pi_{ij}}$ of a direction $\overrightarrow{P_iP_j}$ on the plane is done by the formula:

$$\overrightarrow{\pi_{ij}} = \overrightarrow{P_iP_j} - \left(\vec{k} \cdot \overrightarrow{P_iP_j}\right) \circ \vec{k} \tag{35}$$

where $\cdot$ is the dot product and $\circ$ is the Hadamard product:

$$a \circ b = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \circ \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_1b_1 \\ a_2b_2 \\ a_3b_3 \end{pmatrix} \tag{36}$$

Fig. 5.7 shows an example of how the edges are projected onto the image plane.

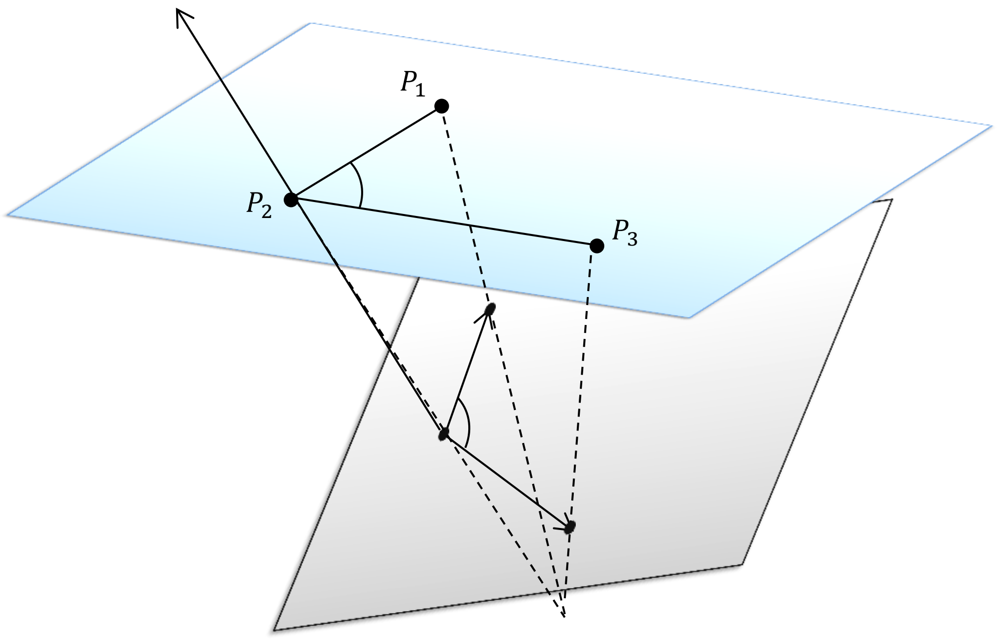


Figure 5.7 - A sample of the projection of two edges on the image plane

Then, in order to obtain the normal $\vec{k}$, a non-linear system is solved, in which is imposed that all the four angles formed by the projections of the edges correspond to those computed on the image:

$$\begin{cases} \cos\alpha_1 = \dfrac{\overrightarrow{\pi_{12}} \cdot \overrightarrow{\pi_{14}}}{\|\overrightarrow{\pi_{12}}\|\|\overrightarrow{\pi_{14}}\|} \\ \cos\alpha_2 = \dfrac{\overrightarrow{\pi_{23}} \cdot \overrightarrow{\pi_{21}}}{\|\overrightarrow{\pi_{23}}\|\|\overrightarrow{\pi_{21}}\|} \\ \cos\alpha_3 = \dfrac{\overrightarrow{\pi_{34}} \cdot \overrightarrow{\pi_{32}}}{\|\overrightarrow{\pi_{34}}\|\|\overrightarrow{\pi_{32}}\|} \\ \cos\alpha_4 = \dfrac{\overrightarrow{\pi_{41}} \cdot \overrightarrow{\pi_{43}}}{\|\overrightarrow{\pi_{41}}\|\|\overrightarrow{\pi_{43}}\|} \end{cases} \tag{37}$$

In place of each direction $\overrightarrow{\pi_{ij}}$ is put the formula (35). The vector $\vec{k}$ is the solution of this non-linear system. It corresponds to the direction of the z-axis of the camera.

5.2.2. Finding the x-axis

The next step consists of finding the camera x-axis from the camera z-axis and considering the projection of the polygon on the image.

Certainly, the x-axis is orthogonal to the z-axis, but this condition is not sufficient in order to find the correct direction. This gives the information about the direction in which the camera is pointing, but gives no information on how the camera is rotated around its optical axis.

As shown in Fig. 5.8, the rotation around the z-axis can be inferred observing the image that results from projection.

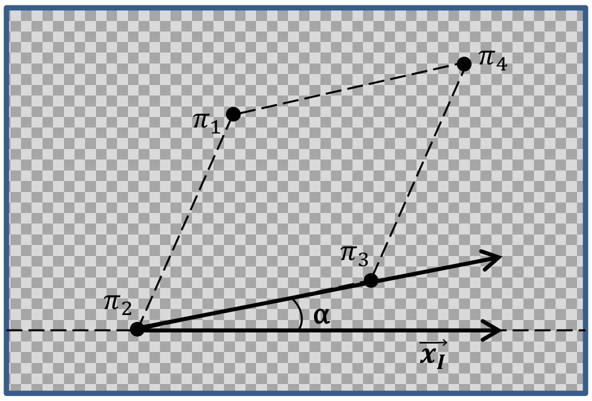


Figure 5.8 - The angle α is measured between the bottom edge of the image, with direction $\overrightarrow{x_I}$, and an edge of the polygon

Choosing an arbitrary side of the polygon, for example $\vec{s} = \overrightarrow{\pi_2\pi_3}$ and the horizontal direction $\overrightarrow{x_I}$, with components (1, 0), the cosine of the angle between $\vec{s}$ and $\overrightarrow{x_I}$ is computed as follows.

$$\cos \alpha = \frac{\vec{x_I} \cdot \vec{s}}{\|\vec{x_I}\| \|\vec{s}\|} \tag{38}$$

Then, it is possible to find the direction $\vec{\imath}$ of the camera x-axis by formulating the two following conditions:

1. the $\vec{\imath}$ direction must be orthogonal to the $\vec{k}$ direction, that is the z-axis of the camera.
2. the $\vec{\imath}$ direction must form an angle α with the vector $\overrightarrow{q_2q_3}$, that is the direction projected on the 3D image plane corresponding to the 2D vector $\overrightarrow{\pi_2\pi_3}$ observed on the image.

The second condition is not immediately applicable. In order to have the vector $\overrightarrow{q_2q_3}$, it is necessary to have the points q_2 and q_3. To have the exact equation of the image plane would be ideal, but in this method is not expected to be known, since the focal length is an intrinsic parameter of the camera and, as mentioned, is not calculated. Only the normal $\vec{k}$ of the image plane is known.

However, it is possible to impose that the plane passes through a known point, for example, the P_2 point, that is one of the vertices of the square (Fig. 5.2). The P_2 is so considered as point q_2. Then a plane having normal $\vec{k}$ and passing through q_2 is obtained. This is not the exact image plane, but surely a plane on which is possible observing the angles as they are effectively projected.

To obtain the q_3 point it is sufficient to intersect the ray of projection $\overline{P_3C_p}$ and the image plane. So the vector $\overrightarrow{q_2q_3}$ to be used in the second condition is available.

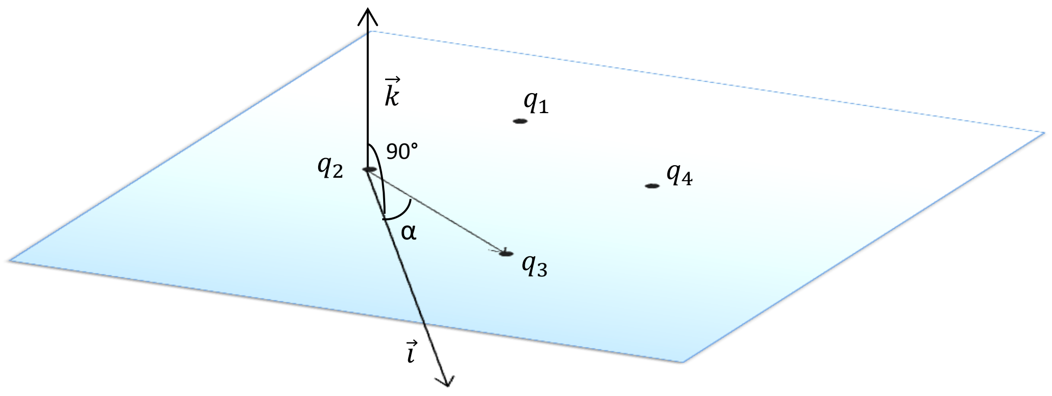


Figure 5.9 – Graphic explanation of the conditions that $\vec{\imath}$ has to respect. It has to form the angle α with $\overrightarrow{q_2q_3}$ and to be orthogonal with $\vec{k}$

The conditions mentioned above are included in a non-linear system. The first condition is formulated as:

$$\vec{k} \cdot \vec{\imath} = 0 \tag{39}$$

The second condition is formulated as:

$$\frac{\vec{\imath} \cdot \overrightarrow{q_2 q_3}}{||\vec{\imath}||\,||\overrightarrow{q_2 q_3}||} = \cos\alpha \tag{40}$$

So, the system to be solved is

$$\begin{cases} \vec{k} \cdot \vec{\imath} = 0 \\ \vec{\imath} \cdot \overrightarrow{q_2 q_3} - \cos\alpha\,(||\vec{\imath}||\,||\overrightarrow{q_2 q_3}||) = 0 \end{cases} \tag{41}$$

The solution $\vec{\imath}$ is the direction of the x-axis of the camera.

5.2.3. Finding the y-axis

The last axis to be found is the $\vec{\jmath}$ direction, which is the y-axis of the camera. This direction is easily obtained by applying the cross product, as follows:

$$\vec{\jmath} = \vec{k} \times \vec{\imath} \tag{42}$$

The vectors $\vec{\imath},\ \vec{\jmath},\ \vec{k}$ are the columns that form the rotation matrix. But a fundamental property of the rotation matrix is that the norm of its columns is equal to 1.

Then, the vectors $\vec{\imath},\ \vec{\jmath}$ and $\vec{k}$ have to be normalized, so as they become unit vectors:

$$\hat{\imath} = \frac{\vec{\imath}}{||\vec{\imath}||}, \qquad \hat{\jmath} = \frac{\vec{\jmath}}{||\vec{\jmath}||}, \qquad \hat{k} = \frac{\vec{k}}{||\vec{k}||} \tag{43}$$

Given the components of the unit vectors:

$$\hat{\imath} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}, \qquad \hat{\jmath} = \begin{pmatrix} x_j \\ y_j \\ z_j \end{pmatrix}, \qquad \hat{k} = \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix} \tag{44}$$

The rotation matrix from the robot A (or B) to the camera C is

$$(R_A)_C = (\hat{\imath} \quad \hat{\jmath} \quad \hat{k}) = \begin{pmatrix} x_i & x_j & x_k \\ y_i & y_j & y_k \\ z_i & z_j & z_k \end{pmatrix} \tag{45}$$

5.3. Transformation matrices

The third stage of the alternative method consists of using the results of the camera extrinsic calibration in order to obtain the robot calibration.

Considering that in the second stage the translations and rotations of the robots A and B with respect to the camera C have been obtained. $(R_A)_C$ and $(t_A)_C$ are the rotation and the translation from A with respect to C. $(R_B)_C$ and $(t_B)_C$ are the rotation and the translation from B with respect to C. Then, the transformation matrices 4-by-4 have the following structure:

$$(T_A)_C = \begin{pmatrix} (R_A)_C & (t_A)_C \\ 0_3 & 1 \end{pmatrix}, \qquad (T_B)_C = \begin{pmatrix} (R_B)_C & (t_B)_C \\ 0_3 & 1 \end{pmatrix} \tag{46}$$

The final operation consists of applying the formula that allows obtaining the position from robot A with respect to robot B:

$$(T_A)_B = (T_A)_C * {(T_B)_C}^{-1} \tag{47}$$

Having a point P_A expressed in the coordinates system of robot A, the transformation matrix $(T_A)_B$ is used to express the point P_A in the coordinates system of robot B, as P_B:

$$P_B = (T_A)_B * P_A \tag{48}$$

In order to apply this formula, the 3D points are expressed in homogeneous coordinates (with an additional coordinate set to 1), since the transformation matrix is 4-by-4:

$$\begin{pmatrix} X_B \\ Y_B \\ Z_B \\ 1 \end{pmatrix} = (T_A)_B * \begin{pmatrix} X_A \\ Y_A \\ Z_A \\ 1 \end{pmatrix} \tag{49}$$

5.4. Implementing the method with MATLAB

The implementation of the alternative method has been done by means of MATLAB. The choice for the developing language MATLAB/Simulink has been done for its simplicity and clarity as well as for its speed and proven reliability. It has a powerful and intuitive scripting language that has been very useful for the rapid implementation of the algorithms required in this project.

This application has been implemented to calibrate the position of two robots, according to the procedure of the alternative method described in the previous sections. In order to calibrate the position of the camera with respect to each robot, two text files are read, containing the points given by the interface described in Chapter 6.

The extrinsic calibration of the camera is made according to the steps described respectively in sections 5.1, 5.2, 5.3. The algorithm finds, in the following order:

- the coordinates of the optical center with respect to the each robot’s reference frame (translation vectors),

- the axes of the camera reference frame with respect to each robot's reference frame (rotation matrices),
- the 4-by-4 transformation matrix from the camera with respect to each robot
- the 4-by-4 transformation matrix from one robot with respect to the other one.

As shown in Fig. 5.10, the application has been divided into different functions, each one performing one step of the algorithm. The functions are called from a main script called *robotcal.m*.

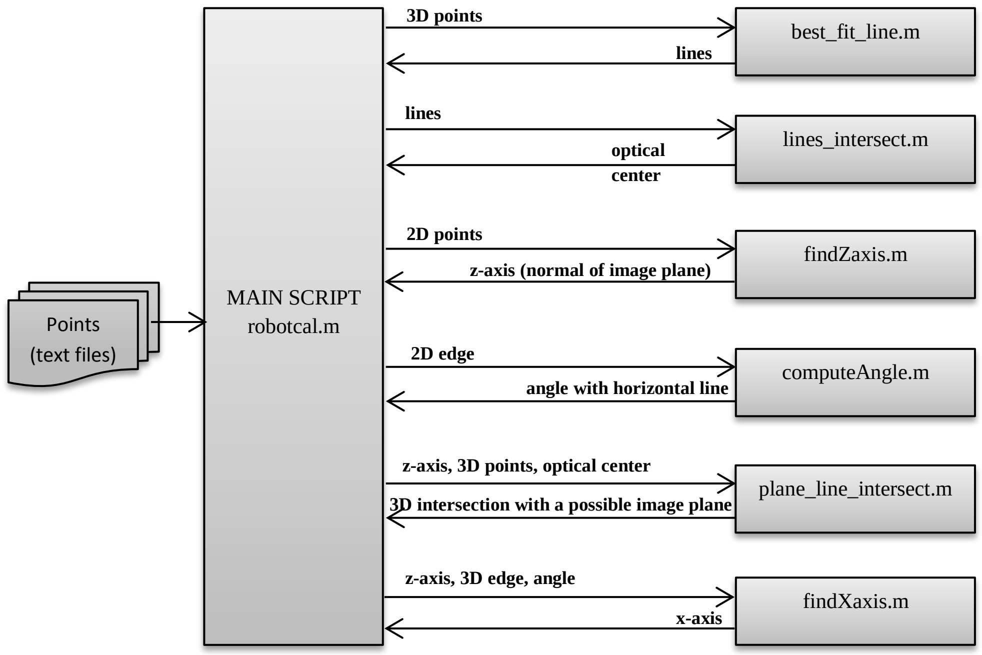


Figure 5.10 - Block diagram of the alternative method in MATLAB.

The procedure is described below with reference to a single robot, being understood that the process is exactly the same for both robots.

5.4.1. Loading of points

The first operation of the main script *robotcal.m* loads and stores into matrices the input text files. The text files are two:

- one of 16 rows that contains 4 points for each of the 4 lines. Each row is a point (X, Y, Z), without the (u, v) projections. The points from 1 to 4 belong to the first line, those from 5 to 8 to the second line, etc…
- one of 4 rows that contains the four points forming the coplanar square. Each row contains 5 values: the 3D coordinates (X, Y, Z) and the (u, v) projection in pixel detected on the image. The points are in a specific order, so that the sequence 1,2,3,4 describe the vertices of the square along the contour counterclockwise.

Then, the 16 points belonging to the four straight lines are stored in a 16x3 matrix called *L*. The 4 points belonging to the square are stored in a 4x5 matrix called *S*.

5.4.2. Finding the optical center

In order to find the optical center, two functions are called:

- ***[p v] = best_fit_line (xs, ys, zs).*** This function is called four times, one for each line. It computes the line of best fit (or the least squares line) with the minimum squared distances from four 3D points. The vectors *xs, ys, zs* are, respectively, the vectors with the x, y and z coordinates of the points contained in *L*. The function returns one point and the normalized direction of the line.
- ***[C] = lines_intersect (ps, vs).*** This function computes the intersection of the four straight lines in the least square sense, that is the point with the minimum squared distance from each line. The matrices *ps* and *vs* contain the points and the normalized direction of the straight lines. The function returns the estimated intersection *C*.

5.4.3. Finding the z-axis of the camera

In order to find the direction $\vec{k}$ of the z-axis of the camera, one function is called:

- ***k = findZaxis (S, C).*** This function computes the angles of the projected polygon. The (u, v) coordinates of the polygon vertices are in the fourth and fifth column of the *S* matrix. After, the non-linear system (36) is prepared and solved, by calling the MATLAB function *fsolve().*

5.4.4. Finding x-axis and y-axis of the camera

In order to obtain the direction $\vec{\imath}$ of the x-axis, three functions are called:

- ***[cs] = computeAngle (2Ded).*** This function computes the cosine of the angle formed by the edge 2Ded (that corresponds to $\overrightarrow{\pi_2\pi_3}$ in Fig. 5.8) and the horizontal direction (1, 0).
- ***[q] = line_plane_intersect (k, C, S).*** This function is used to obtain the projection of the point p_3 on the image plane. The considered image plane is not the real image plane, but a plane with normal $\vec{k}$ passing through the point p_2 of *S*. The point *q* is obtained as intersection of the plane with the line passing through p_3 and the optical center *C*.
- ***[i] = findXaxis (k, cs, 3Ded).*** This function solves a non-linear system by calling the *fsolve()* function. In the linear system there are the conditions (41): the vectors $\vec{\imath}$ and $\vec{k}$ must to be orthogonal (dot product equal to zero) and the direction $\vec{\imath}$ must form the same angle (with cosine *cs*) computed by the ***computeAngle()*** function. The function returns the direction $\vec{\imath}$ of the x-axis of the camera.

In order to find the y-axis, is sufficient to execute the call cross() that compute the cross product of $\vec{k}$ and $\vec{\imath}$.

5.4.5. Transformation matrices and robot calibration

In the last instructions of the main script *robotcal()*, the vectors $\vec{\imath}$, $\vec{\jmath}$, $\vec{k}$ are normalized, then the transformation matrix from the robot with respect to the camera is composed.

Finally, since the operations described above are executed for both robots, the transformation matrix from one robot to the other one is computed applying the formula (47).

6. ACQUIRING POINTS WITH OPENCV

The methods described in the previous chapters make use of an interface, which allows detecting a set of 3D points with the corresponding pixel coordinates of their projections.

Since the work has been done by using the instrumentation described in Chapter 2, the program works properly under the following conditions:

1. The interface communicate with the camera used in the laboratory (Section 2.4), hence the camera drivers have to be installed on PC and included into the C++ source code. Another camera could be used in order to acquire the images, but the code should be modified.
2. It is assumed that an IR LED (Fig. 2.4) is placed as end-effector of the robot arm in order to highlight the position of the points used along the calibration process. Accordingly, an infrared filter is placed on the camera, so that in the image the light is clearly distinguished as bright spot on a dark background (Fig. 6.3).

The aim of this interface (Fig. 6.3) is to capture an unlimited number of frames from the camera and to save the projections in pixel of the points on which the robot arm is moved. The position taken by robot arm is determined by finding the position of the IR light on image. In order to extract the required coordinates (u, v), the image is converted from color to grayscale and segmented, so that the light is separated by the background. The central pixel (u,v) of the light is then computed (as explained below) and finally saved in a text file.

The diagram of Fig. 6.1 shows the whole procedure followed by the program. By clicking on the "Start" button the acquisition of the frames begins. The screen projects a new frame every 25 milliseconds.

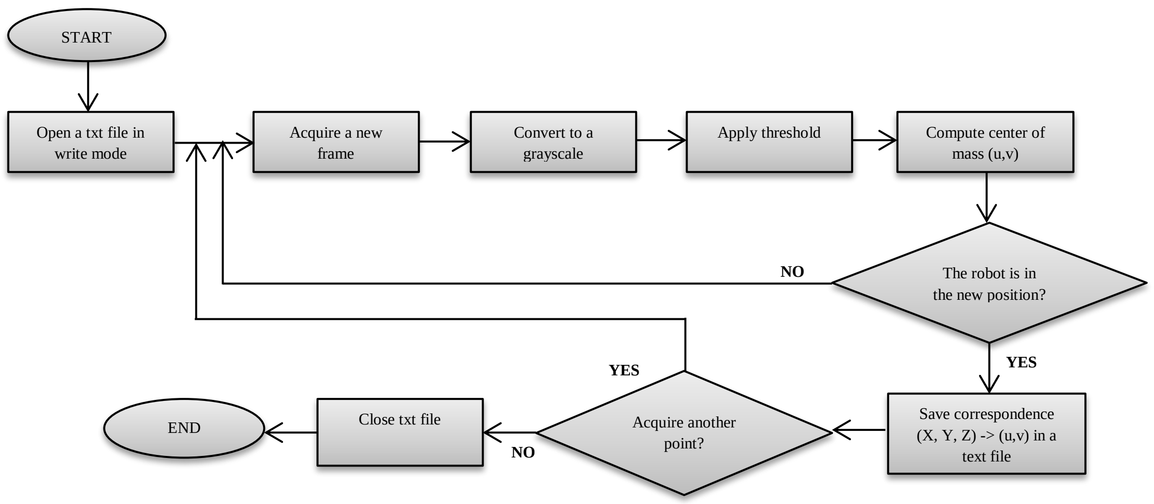
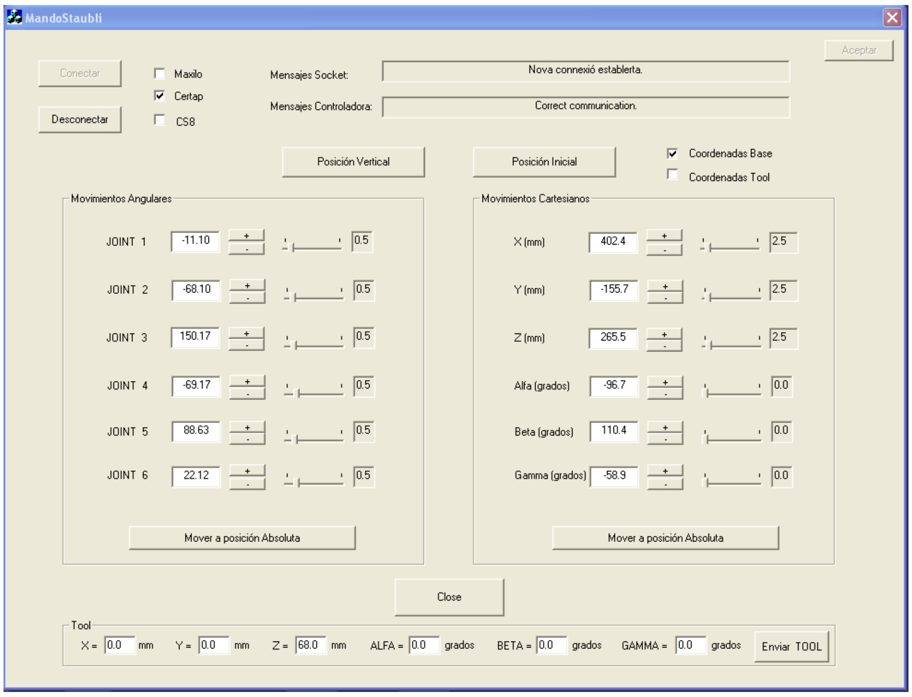

START
Open a txt file in write mode
Acquire a new frame
Convert to a grayscale
Apply threshold
Compute center of mass (u,v)
The robot is in the new position?
NO
YES
Save correspondence (X, Y, Z) -> (u,v) in a text file
Acquire another point?
YES
NO
Close txt file
END

Figure 6.1 - Flow diagram of the program used to acquire the points

The robot is remote-controlled by using a TCP/IP connection, through a Visual Studio interface already available in the PC of laboratory. As shown in Fig. 6.2, this interface allows moving the robot in different ways: it is possible by setting its Cartesian coordinates or by setting the angles of its joints. It is also possible to set the position in an incremental or absolute way. These options are useful during the process of acquisition, in order to put the IR LED in face of the camera, when it is possible.



☐ Maxilo
☑ Certap
☐ CS8

Mensajes Socket: Nova connexió establerta.
Mensajes Controladora: Correct communication.

☑ Coordenadas Base
☐ Coordenadas Tool

Movimientos Angulares

JOINT 1	-11.10	0.5
JOINT 2	-68.10	0.5
JOINT 3	150.17	0.5
JOINT 4	-69.17	0.5
JOINT 5	88.63	0.5
JOINT 6	22.12	0.5

Movimientos Cartesianos

X (mm)	402.4	2.5
Y (mm)	-155.7	2.5
Z (mm)	265.5	2.5
Alfa (grados)	-96.7	0.0
Beta (grados)	110.4	0.0
Gamma (grados)	-58.9	0.0

Tool
X = 0.0 mm Y = 0.0 mm Z = 68.0 mm ALFA = 0.0 grados BETA = 0.0 grados GAMMA = 0.0 grados

Figure 6.2 – Image of the interface used to move the robots.

When the robot is moved to a new position, by clicking the “Stop” button it is possible to freeze the acquisition of frames and process the image in order to find the projection of the highlighted point. Then, the (u, v) coordinates are instantly saved on the output text file with the corresponding 3D coordinates (X, Y, Z).

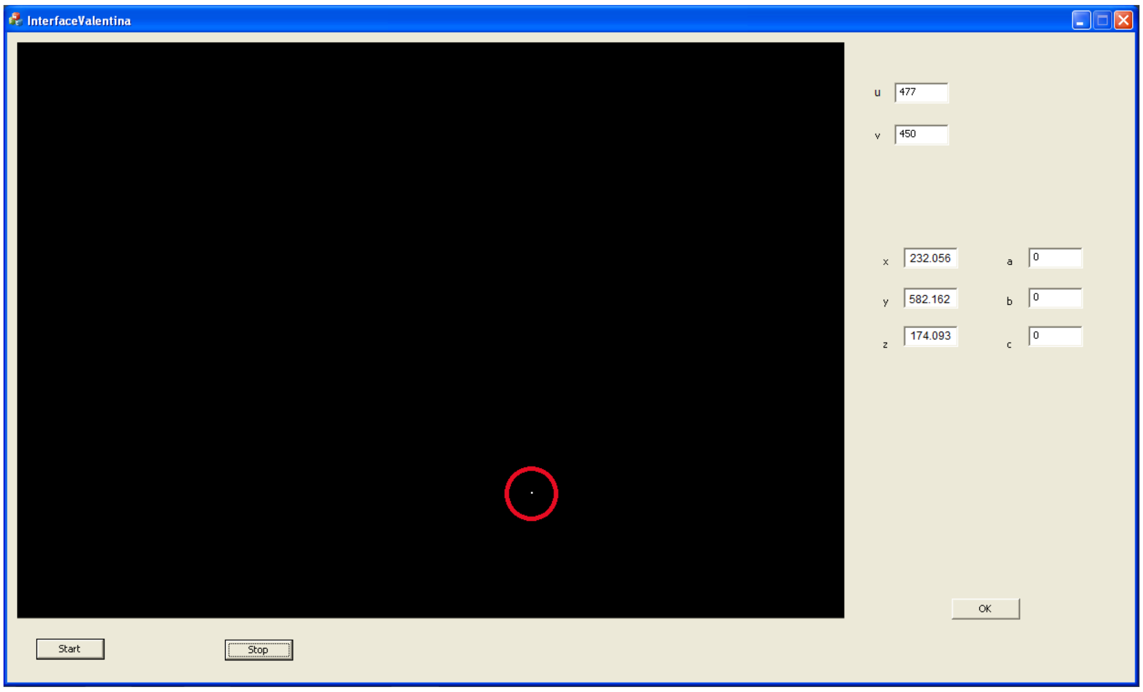
The OpenCV class used to work on images is *IplImage*. On the IplImage objects it is possible to call several image processing functions, as the ones used in this project:

1. conversion of the image from color into grayscale, by calling the *cvCvtColor()* function;
2. image binarization, based on a threshold, in order to isolate and separate the light from the rest of image, by calling the *cvThreshold()* function;
3. calculus of the center of mass, that is the average point of the white pixels of the image.

The conversion of the image from color into grayscale is made in order to segment the image, separating the part with high intensity level from the one with low intensity. The intensity level, in an image, is the measure of the pixel luminosity. It has a range (0-255), where 0 is the absence of luminosity (black) and 255 is the maximum level of luminosity (white).

The segmentation consists of converting the image in a new one that has only two intensity levels: black (0) and white (1). The pixels are set to 0 if their intensity level is lower than a certain threshold, are set to 1 otherwise. This operation allows to well separate the pixels that belongs to the IR light from the other pixels. The threshold has chosen to be 100.

On the binary image that has be obtained, the center of mass is computed. It is the pixel having average coordinates among all white pixels. It corresponds to the central coordinates of the IR light, then to the projection (u, v) of the 3D point on which the LED is moved.



u 477
v 450
x 232.056
y 582.162
z 174.093
a 0
b 0
c 0

Figure 6.3 - Screenshot of the interface used to acquire the points

The center of mass is obtained by computing a set of particular weighted averages of the image pixels intensities, named geometric moments. For the calculus are only used the moment of order 0 (that corresponds to the area of the white region in pixels) and the moments of order 1, as explained below.

The geometric moments are computed through the *cvMoments()* function, with the following prototype:

```
void cvMoments(

        const CvArr*     image,
        CvMoments*       moments,
        int              isBinary = 0

)
```

By calling this function, all the image moments are computed at once. The first argument is the image (it is possible to give as first parameter an IplImage object); the second arguments is an output on which the function stores the computed moments; the last argument is set to 1 if the image is binary.

A pixel of the binary image, as explained above, belongs to the light if its value is 1, to the background if the value is 0. The value of a pixel can be considered the output of a function f(x,y), where x and y are the coordinates of one pixel.

$$f(x,y) = \begin{cases} 1, & if\ (x,y) \in light \\ 0, & if\ (x,y) \notin light \end{cases}$$

Then, the geometric moment of order (p+q) is defined as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y)$$

In this particular case, the moments considered are those of first order (where p+q=1), which allows to compute the centroid coordinates (the center of mass of the light). These moments are:

$$m_{10} = \sum_x \sum_y x f(x,y)$$

$$m_{01} = \sum_x \sum_y y f(x,y)$$

In order to compute the center of mass the moment of order 0 is also considered, that is the light area in pixels:

$$m_{00} = \sum_x \sum_y f(x,y)$$

The center of mass of the light, then the projection of the 3D point on which the infrared light has origin, is (u, v):

$$u = \frac{m_{10}}{m_{00}}, \qquad v = \frac{m_{01}}{m_{00}}$$

Then, the output text file of this interface is written placing in each row 5 space-separated values, like the following example:

```
413.7 245.0 154.1 257 289
253.7 245.0 154.1 46 291
253.7 245.0 314.1 62 500
413.7 245.0 314.1 264 515
416.2 280.0 154.1 256 290
271.2 280.0 154.1 45 291
271.2 280.0 299.1 61 500
416.2 280.0 299.1 263 515
```

For each row, the first three values are the (X, Y, Z) coordinates of the 3D point. The other two values are the (u, v) coordinates in pixel of its projection.

For each robot it has been followed the procedure just described, then two text file have been produced through this interface. Each one contains the acquired 3D points with the corresponding projections on the image. These files have been used as input for both reference and alternative method.

7. RESULTS

In this chapter, the results of camera calibration and robot calibration are shown and discussed. First, all the matrices returned by reference and alternative method, which contains the intrinsic and the extrinsic parameters, are exposed.

This chapter also shows the results of practical application of both methodologies, which have been tested and compared. Tests are based on comparing some “real positions”, which are given by robot B with respect to its reference frame, to the “expected positions” given by applying the transformation matrices to the positions expressed with respect to the reference frame of robot A. The accuracy is measured by means of the distance between real positions and expected ones.

7.1. Camera calibration results

The methods explained in Chapters 4 and 5 show how to calibrate the position of one robot with respect to the other one, through the computation of several parameters which express the relationships between three objects of the work environment: the camera and the two robots A and B. These objects are placed in the laboratory as shown in Fig. 7.1.

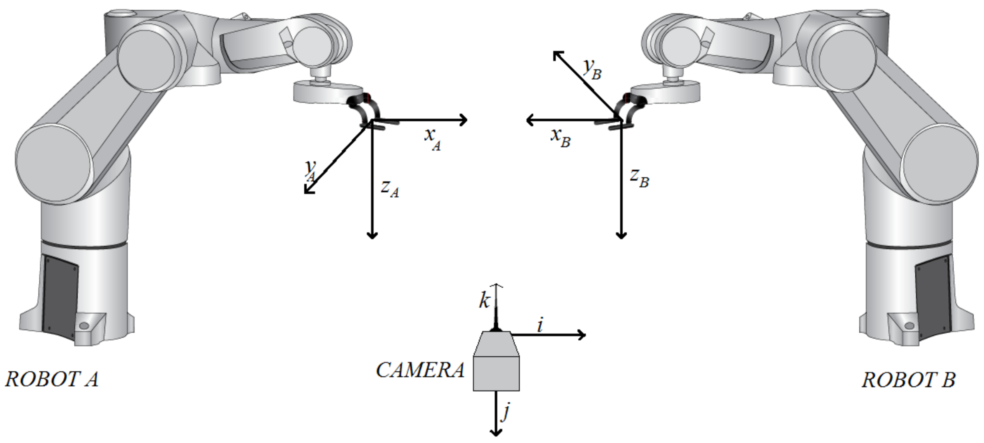


Figure 7.1 – The components of the system with their reference frames

The camera is set in order to point one spot in the middle of the two robots. The robot arms are in opposite positions one respect to the other. The objective of the methods was localize the robot A with respect to the robot B. This means that a relationship between the coordinates systems of both robots had to be found, that, given any position expressed in the reference frame of robot A, allows to convert it in the reference frame of robot B.

This relationship is expressed through a translation and a rotation. Through the translation, it is possible to know the off sets between the positions of the robots, along each axis. Through the rotation, it is possible to know how the reference frame of one robot is oriented with respect to the other robot. In order to obtain rotation and translation from one robot with respect to the other one, has been necessary to know rotation and translation from the camera with respect to each robot. In the reference method has been also necessary to find the internal features of the camera.

Then, the experimented methods have required that the parameters had been computed in the following order:

- the intrinsic parameters of the camera (only for reference method), forming a 3-by-4 matrix named $\boldsymbol{I}$ (Section 3.2.3), and an 1-by-5 distortion coefficients vector named $\boldsymbol{d}$ (Section 3.2.4),
- the extrinsic parameters of the camera (Section 3.4) with respect to robot A and robot B, split in translation vectors $(\boldsymbol{t_A})_{\boldsymbol{C}}$ and $(\boldsymbol{t_B})_{\boldsymbol{C}}$ and rotation matrices $(\boldsymbol{R_A})_{\boldsymbol{C}}$ and $(\boldsymbol{R_B})_{\boldsymbol{C}}$.
- the transformation matrices, one from the camera with respect to the robot A, $(\boldsymbol{T_A})_{\boldsymbol{C}}$, and the other from the camera with respect to robot B, $(\boldsymbol{T_B})_{\boldsymbol{C}}$. The transformation matrices are obtained by composing the rotation matrices and the rotation vector, as explained in Section 3.4.3.
- the transformation matrix $(\boldsymbol{T_A})_{\boldsymbol{B}}$ from robot A with respect to robot B, by applying the formula (41).

In next sections, the matrices returned by both methods are reported.

7.1.1. Intrinsic calibration of the camera

The first results obtained in the reference method have been the intrinsic parameters of the camera. In order to achieve this goal, 25 images of a chessboard 4-by-5 (Fig. 4.2) have been captured. These images have been given in input to the Camera Calibration Toolbox (Section 4.1). Through the information taken from each image, the MATLAB tool solved a linear system and then returned the following values:

Focal length:	**fc = [576.20944 585.74690] ± [20.82911 23.11918]**
Principal point:	**cc = [355.47462 289.99393] ± [18.46172 23.36451]**
Skew:	**alpha_c = [0.00000] ± [0.00000] => angle of pixel axes = 90.00000 ± 0.00000 degrees**
Distortion:	**kc = [-0.38212 0.31751 0.01003 -0.00160 0.00000] ± [0.08960 0.21657 0.01414 0.00830 0.00000]**
Pixel error:	**err = [0.36357 0.39688]**

The pixel error can be considered as an overall measure of the accuracy of the intrinsic parameters. The error values in horizontal direction and in vertical direction are respectively estimated as 0.36357 and 0.39688 pixels. They are less than 0.5, as it was expected, then the results can be considered acceptable.

According to the matricial notation described in Section 3.2.3, the matrix of intrinsic parameters with the above values is:

$$I = \begin{pmatrix} 576.20944 & 0 & 355.47462 & 0 \\ 0 & 585.74690 & 289.99393 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The 1-by-5 vector of distortion coefficients is:

$$d = (-0.38212 \quad 0.31751 \quad 0.01003 \quad -0.00160 \quad 0.00000)$$

where the first three elements belong to radial distortion and the remaining two belong to tangential distortion (Section 3.2.4).

7.1.2. Extrinsic calibration of the camera with reference method

In Sections 7.1.2 and 7.2.2, the position and the orientation of the camera with respect to the robots are reported, by means of their corresponding translation vectors and rotation matrices.

In the reference method, these parameters have been obtained by using the Visual Studio interface described in Section 4.4. The interface has required as input the intrinsic parameters of the camera and, for each robot, a set of 3D points with their projections.

Once all the needed inputs have been loaded through the interface shown in Fig. 4.7, the results shown below have been returned.

The **translation vectors** contain the coordinates of optical center, expressed in millimeters, with respect to the reference frame of each robot.

The translation vector from the camera with respect to robot A is

$$(t_A)_C = \begin{pmatrix} 452.36 \\ 880.86 \\ 33.69 \end{pmatrix}$$

The translation vector from camera with respect to the robot B is

$$(t_B)_C = \begin{pmatrix} 516.13 \\ -793.45 \\ 34.38 \end{pmatrix}$$

The **rotation matrices** express the orientation of the camera with respect to the robots reference frames.

The rotation matrices from the camera with respect to the robots A and B are:

$$(R_A)_C = \begin{pmatrix} 0.9996 & -0.0240 & -0.0116 \\ -0.0107 & 0.0357 & -0.9993 \\ 0.0245 & 0.9991 & 0.0354 \end{pmatrix}$$

$$(R_B)_C = \begin{pmatrix} -0.9993 & 0.0270 & 0.0239 \\ 0.0232 & -0.0279 & 0.9993 \\ 0.0276 & 0.9992 & 0.0272 \end{pmatrix}$$

The **transformation matrices**, as shown in (46), contains both rotation matrices and translation vectors. The transformation matrices from the robot A and B with respect to the camera are:

$$(T_A)_C = \begin{pmatrix} 0.9996 & -0.0240 & -0.0116 & 452.36 \\ -0.0107 & 0.0357 & -0.9993 & 880.86 \\ 0.0245 & 0.9991 & 0.0354 & 33.69 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

:

$$(T_B)_C = \begin{pmatrix} -0.9993 & 0.0270 & 0.0239 & 516.13 \\ 0.0232 & -0.0279 & 0.9993 & -793.45 \\ 0.0276 & 0.9992 & 0.0272 & 34.38 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

7.1.3. Extrinsic calibration of the camera with alternative method

In the alternative method (Chapter 5), the extrinsic parameters of the camera with respect to each robot are obtained without need of intrinsic parameters. The alternative method has been run through a MATLAB script described in Section 5.4. It has taken in input the 3D points arranged according to the second configuration explained in Section 5.1.2.

Translation vectors from robot A and B with respect to the camera:

$$(t_A)_C = \begin{pmatrix} 452.68 \\ 878.19 \\ 34.48 \end{pmatrix}$$

$$(t_B)_C = \begin{pmatrix} 513.76 \\ -781.92 \\ 33.84 \end{pmatrix}$$

Rotation matrices from robot A and B with respect to the camera:

$$(R_A)_C = \begin{pmatrix} 0.9998 & -0.0215 & 0.0000 \\ -0.0000 & 0.0000 & -1.0000 \\ 0.0215 & 0.9998 & 0.0000 \end{pmatrix}$$

$$(R_B)_C = \begin{pmatrix} -0.9999 & -0.0164 & 0.0000 \\ 0.0000 & -0.0000 & 1.0000 \\ -0.0164 & 0.9999 & 0.0000 \end{pmatrix}$$

Resulting transformation matrices from robot A and B with respect to the camera:

$$(T_A)_C = \begin{pmatrix} 0.9998 & -0.0215 & 0.0000 & 452.68 \\ -0.0000 & 0.0000 & -1.0000 & 878.19 \\ 0.0215 & 0.9998 & 0.0000 & 34.48 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$(T_B)_C = \begin{pmatrix} -0.9999 & -0.0164 & 0.0000 & 513.76 \\ 0.0000 & -0.0000 & 1.0000 & -781.92 \\ -0.0164 & 0.9999 & 0.0000 & 33.84 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Though it may seem that results of the alternative method are quite close to results of the reference method, the differences are relevant, and this is reflected by sensible differences in the transformation matrices respectively obtained (Section 7.2).

7.2. Robot calibration results

According to the formula of coordinates system change detailed in Section 3.4.3 (see also Fig. 4.6), the matrix which relates the position of robot A with respect to robot B is obtained from the transformation matrices above exposed. Two different results will be achieved, depending on the method used to calibrate the camera.

7.2.1. Reference method

The robot calibration through the reference method has given the following result:

$$(T_A)_B = \begin{pmatrix} -0.9999 & 0.0122 & 0.0033 & 978.04 \\ -0.0122 & -0.9999 & 0.0081 & 93.51 \\ 0.0034 & 0.0081 & 0.99996 & 3.98 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

7.2.2. Alternative method

The robot calibration through the alternative method has given the following result:

$$(T_A)_B = \begin{pmatrix} -0.9993 & 0.0000 & -0.0379 & 967.35 \\ -0.0000 & -1.0000 & 0.0000 & 96.27 \\ -0.0379 & -0.0000 & 0.9993 & 20.14 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The transformation matrices obtained have important differences that can be particularly noted in the translation vectors. In order to evaluate which method gives better results, it is necessary to make accuracy tests, as explained in 7.3.3.

7.3. Validation of the methods

In this section, the comparison and accuracy testing of both methods are described. The section starts with description of the manual method used in order to obtain, having some coordinates in the reference frame of robot A, the coordinates of the same positions in the reference frame of robot B, which are named “real positions”. In other words, the sharp tip end effectors (Fig. 2.3) of both robots are placed at the same positions (reaching contact) and their coordinates are taken in their own reference frames.

The section also shows how to compare the expected positions given by both methods, and finally how to estimate the accuracies of these positions by comparing them with the “real positions”.

7.3.1. Getting real positions

In order to identify coincident points in the space, two sharp tips have been used as end effectors (Fig. 2.3). One of these sharp-tips has been fitted with a flexible part in order to avoid collision between rigid parts.

The arm of the robot A has been moved to predefined positions. For each position, the arm of the robot B has been moved until the tips of both end effectors coincided in space. Then, the robots have provided the position of the same point with respect to their own reference systems. This position is expressed as $P_i^A = (X_i^A, Y_i^A, Z_i^A)$ in the reference frame of robot A, and as $P_i^B = \left(X_i^B, Y_i^B, Z_i^B\right)$ in the reference frame of robot B.

Nine points have been considered, as shown in Table 7.1. The points P_i^A of the robot A have been arbitrarily chosen, so that they were representative and not concentrated on a single area. Each point P_i^B, obtained through the procedure described above, has the same real position of the corresponding P_i^A point.

ROBOT A				ROBOT B			
	X	Y	Z		X	Y	Z
P_1^A	529.6	346	103	P_1^B	446	-250.9	99.5
P_2^A	515.1	203.8	202	P_2^B	463.5	-108.9	198.5
P_3^A	610.1	175	47	P_3^B	368.5	-78.9	43.5
P_4^A	490.1	40	-63	P_4^B	491	54.1	-65.5
P_5^A	545.1	203.8	142	P_5^B	433.5	-108.9	138.5
P_6^A	576.1	204.8	202	P_6^B	403.5	-108.9	198.5
P_7^A	573.6	247.2	185.6	P_7^B	402.7	-151.4	181
P_8^A	427.3	258.6	94.1	P_8^B	550.2	-166.4	91
P_9^A	538.6	184.7	203.1	P_9^B	440.2	-88.9	198.5

Table 7.1 – Arbitrary positions P_i^A and corresponding "real positions" P_i^B obtained by using the sharp-tip end effector

7.3.2. Comparison between the methods

In this section, a comparison between the methods is made. The idea consists of obtaining the coordinates of the points with respect to robot B reference frame, by applying the transformation matrices (of both methods) to the P_i^A points expressed with respect to robot A reference frame.

The transformation matrix $(T_A)_B$ computed by the reference method is named T^{OCV} and the one computed by the alternative method is named T^{MAT}.

Then, for each point P_i^A the following computations have been made:

$$P_i^{OCV} = T^{OCV} * P_i^A$$

$$P_i^{MAT} = T^{MAT} * P_i^A$$

In table 7.2, the points $P_i^{OCV}$and P_i^{MAT} computed by means of both methods are reported.

POINT	**REFERENCE METHOD (P_i^{OCV})**			**ALTERNATIVE METHOD (P_i^{MAT})**		
	X	Y	Z	X	Y	Z
1	453.0544	-258.0775	111.5766	434.2170	-249.7300	102.9960
2	466.1448	-114.9129	209.3656	444.9547	-107.5300	202.4763
3	370.2915	-88.5303	54.4708	355.8957	-78.7300	43.9843
4	488.2695	47.0291	-57.0196	479.9807	56.2700	-61.3906
5	435.9498	-115.7649	149.4736	417.2497	-107.5300	141.3813
6	405.1631	-116.6570	209.5811	383.9974	-108.5300	200.1644
7	408.1260	-159.1551	193.5177	387.1172	-150.9300	183.8706
8	554.2485	-169.5102	101.6217	536.7827	-162.3300	97.9794
9	442.4178	-96.0926	210.3907	421.4295	-88.4300	202.6848

Table 7.2 – Coordinate of the expected points computed by reference method and alternative method

In order to have a measure of the differences between the expected points computed by the methods, is useful to compute the distance between them. Given two points, P_i^{OCV} and P_i^{MAT}, the Euclidean distance can be measured:

$$d(P_i^{OCV}, P_i^{MAT}) = \left\|\overline{P_i^{OCV} P_i^{MAT}}\right\| = \sqrt[2]{\left(X_i^{OCV} - X_i^{MAT}\right)^2 + \left(Y_i^{OCV} - Y_i^{MAT}\right)^2 + \left(Z_i^{OCV} - Z_i^{MAT}\right)^2}$$

In table 7.3, for each point the differences along the axes, expressed in absolute value, and the distances are reported.

POINT	**DIFFERENCES**			**DISTANCE** $d(P_{OCV}, P_{MAT})$
	X	Y	Z	
1	18.837440	8.347520	8.580580	22.319422
2	21.190100	7.382940	6.889310	23.473192
3	14.395740	9.800320	10.486530	20.328574
4	8.288740	9.240820	4.371030	13.160618
5	18.700100	8.234940	8.092310	21.977112
6	21.165700	8.127040	9.416710	24.550154
7	21.008800	8.225140	9.647060	24.537489
8	17.465860	7.180290	3.642310	19.232245
9	20.988300	7.662640	7.705810	23.634811

Table 7.3 - Differences along the axes and distances between the results of the methods

The differences between the two method is high, since the results computed have an average distance of about 2 centimeters. Although this is not an exhaustive information about goodness of the methods in order to see which is better. The accuracy testing, explained in next section, is indeed relevant for these purpose.

7.3.3. Accuracy evaluation

In this section is shown how the accuracy is evaluated. It has to be computed by comparing the "expected positions" P_i^{OCV} and P_i^{MAT} obtained with the methods (as shown in 7.3.2) to the "real positions" P_i^B, expressed in the robot B reference frame and obtained through the procedure explained in 7.3.1.

The accuracy of each method for each point is computed as difference in absolute value along each axis and as Euclidean distance d_i^{OCV} (or d_i^{MAT}) between the points.

$$d_i^{OCV} = d(P_i^{OCV}, P_i^B)$$

$$d_i^{MAT} = d(P_i^{MAT}, P_i^B)$$

Finally, the average distance (shown in table 7.4) between its resulting points and the real ones.

POINTS	**REFERENCE METHOD**			**ALTERNATIVE METHOD**		
	X	Y	Z	X	Y	Z
1	7.054460	7.177520	12.076640	11.782980	1.170000	3.496060
	d_1^{OCV}=15.720287			d_1^{MAT}=12.346253		
2	2.644870	6.012940	10.865620	18.545230	1.370000	3.976310
	d_2^{OCV}=12.696948			d_2^{MAT}=19.016138		
3	1.791510	9.630320	10.970840	12.604230	0.170000	0.484310
	d_3^{OCV}=14.707546			d_3^{MAT}=12.614677		
4	2.730490	7.070820	8.480340	11.019230	2.170000	4.109310
	d_4^{OCV}=11.374016			d_4^{MAT}=11.959045		
5	2.449870	6.864940	10.973620	16.250230	1.370000	2.881310
	d_5^{OCV}=13.173823			d_5^{MAT}=16.560460		
6	1.663170	7.757040	11.081120	19.502530	0.370000	1.664410
	d_6^{OCV}13.628244			d_6^{MAT}=19.576921		
7	5.426080	7.755140	12.517700	15.582720	0.470000	2.870640
	d_7^{OCV}=15.693226			d_7^{MAT}=15.851897		
8	4.048580	3.110290	10.621770	13.417280	4.070000	6.979460
	d_8^{OCV}=11.785029			d_8^{MAT}=15.662093		
9	2.217830	7.192640	11.890700	18.770470	0.470000	4.184890
	d_9^{OCV}=14.072725			d_9^{MAT}=19.237067		
AVG	**3.3363**	**6.9524**	**11.0532**	**15.2750**	**1.2922**	**3.4052**
	d_{AVG}^{OCV}=**13.6502**			d_{AVG}^{MAT}=**15.8694**		

Table 7.4 – Results of accuracy tests.

In Fig. 7.1, accuracy evaluation is graphically shown, including the average differences along each axis and the average accuracy (the average distance between the points computed by each method and the real ones).

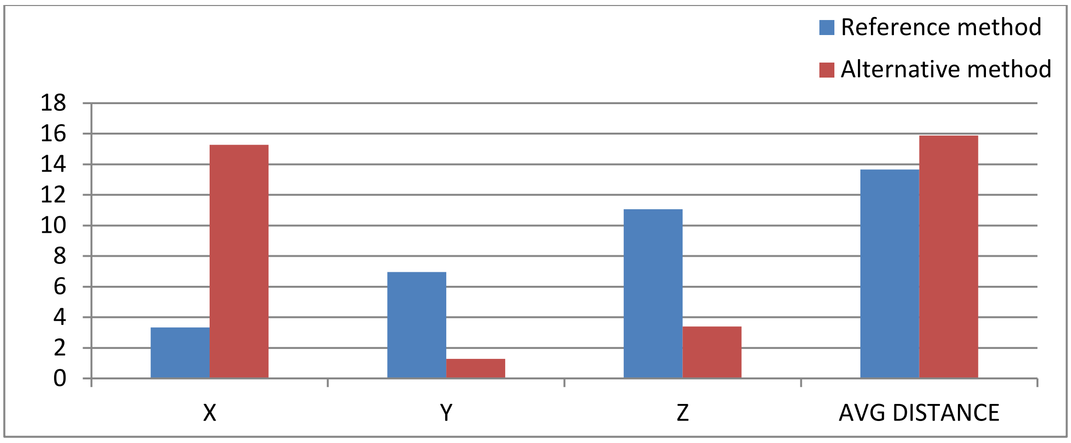


Figure 7.2 – Average differences in absolute value between the results of each methods and the real positions along each axis (X, Y, Z) and average Euclidean distances

Observing on the graph the comparison of the average distances, the reference method (average accuracy d_{AVG}^{OCV}=**13.6502**) is globally more accurate than the alternative method (average accuracy d_{AVG}^{MAT}=**15.8694**).

7.4. Comments about results

The average error committed by the methods was expected to be < 2.5 mm in order to be considered just acceptable in a context of surgery. However, the results show much larger discrepancies, as reported above.

The error seems to have its origin by extrinsic calibration of the camera. The reference and the alternative method give transformation matrices from camera with respect to the robots with relevant differences. In detail, in Sections 7.1.2 and 7.1.3 it can be observed that the differences are more relevant between the transformation matrices $(T_B)_C$ than between the transformation matrices $(T_A)_C$. This is probably due to the position of the camera that has allowed taking points more suitable to define the position of robot A. Although the camera is about in a middle position with respect to the robots, the configuration of them is opposite, so common workspace only allowed projecting the LED to a certain area of the image, probably unfavorable to robot B. Unreliability of the calibration of the camera with respect to robot B has certainly affected the overall calibration from the robot A with respect to robot B.

Fig. 7.2 shows that alternative method is inaccurate with respect to the x-axis, probably because the projections of the LED at the same side of the image has jeopardized the computing of the horizontal correspondences between robot B position's and projections.

Anyway, if the accuracy in X had achieved the same levels than in Y and Z, the accuracy wouldn't be satisfactory either. Some reasons that contribute to these discrepancies are explained in the following sections.

7.4.1. Camera resolution

During the acquisition process, an unavoidable error is committed due to the discretization of the world metric measurements through image pixels.

Each pixel covers a range of the order of a few millimeters in space, depending on the distance between the camera and the point acquired. In fact, if the camera moves away from the points, the range increases and the potential error is higher.

Fig. 7.3 shows an example of a camera simply represented by optical center C_P and image plane. If the camera is far enough from the points A and B, their projections are different, but if the camera is closer to the points, their projections on the image plane are discretized in the same pixel.

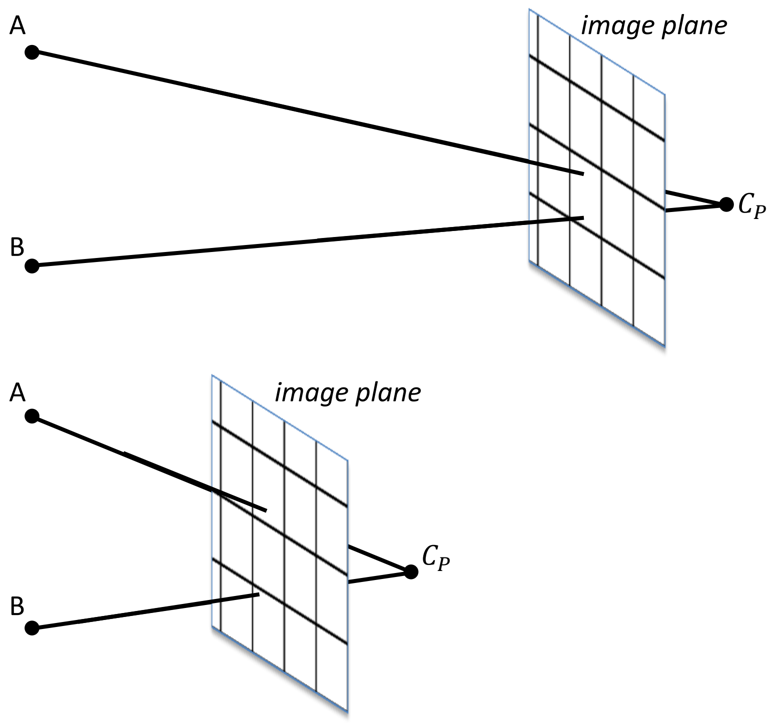


Figure 7.3 - Two cases of a camera focusing two points. In the first case, the points are projected in different pixels, while in the second case the projections are the same.

7.4.2. Point acquisition via IR LED

As described in Chapter 6, in order to acquire the points through the Visual Studio interface, an infrared LED has been used as end-effector on each robot arm.

The main reason of this choice has been to isolate the light and obtain a more reliable segmentation in order to calculate as precisely as possible its center of mass (Chapter 6). Using this approach make it possible to obtain the projection (u, v) of the light ray captured by a camera endowed with IR filter.

The LED has a diameter less than 2 millimeters. It is contained in a nylon stick and the light exit through a hole placed on the top (Fig. 7.4).

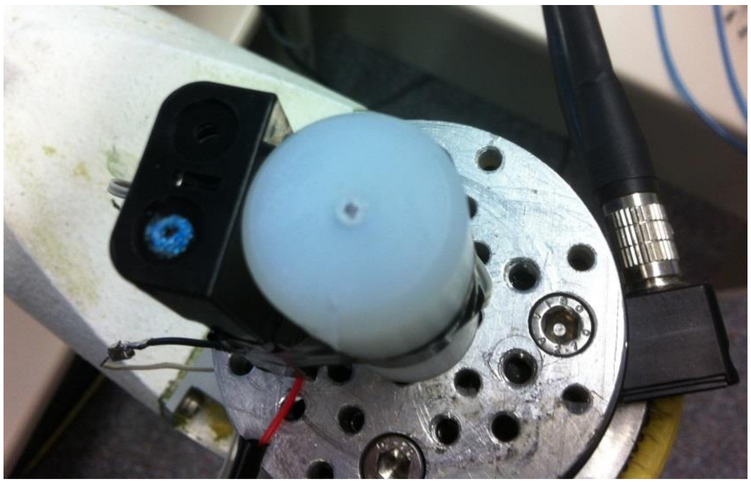
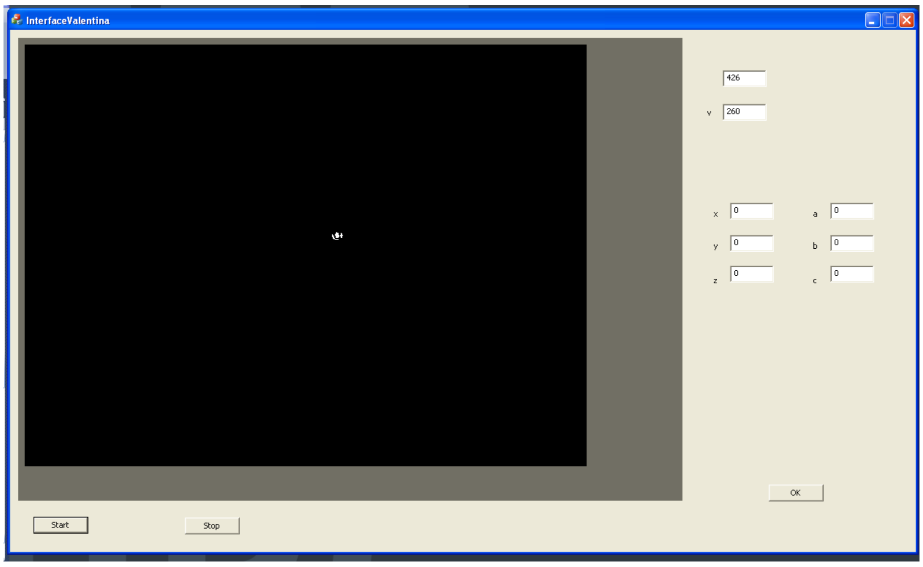


Figure 7.4 - The nylon stick that contains the IR LED.

According to the arm positions the shape of the projected light could result quite different. In fact, it strongly depends on the angle of view of the acquired point and on the position of the LED with respect to the camera (in Fig. 7.5 is shown a bad case). In order to avoid as much as possible these anomalies, the LED should be positioned exactly in front of the camera. Unfortunately this is not always possible. For this reason, the calculation of the center of mass in some cases results unreliable.



426

v 260

x 0 a 0

y 0 b 0

z 0 c 0

Figure 7.5 - An example of irregular light captured by the Visual Studio interface

CONCLUSIONS

This project has shown that the calibration of a camera, and consequently the calibration of the position of two robots by vision, is a procedure far from being immediate.

The manual calibration, usually practiced in the laboratory, requires time and efforts. The adoption of a calibration procedure by vision certainly is an improvement with respect to manual calibration, but it introduces some typical problems about the vision system, described in Section 7.4. According to the results obtained by the calibration method proposed in this thesis, as a first conclusion, it is clear that the objectives declared in Section 1.2 have not been completely achieved.

The errors are clearly unacceptable for surgical employments, since the operations performed in this field generally require a higher level of precision, especially in delicate procedures as orthopedic surgery where the tolerance level is notoriously lower respect to laparoscopic procedures.

In the following paragraphs, some possible improvements have been listed in order to avoid and overcome the detected problems in future works settings.

Improving camera resolution

Two possible solutions could be implemented in order to improve the accuracy in vision system. One option is to move the camera in order to get closer to the positions in which the robot arms are moved. In this way the confluence of points projections in a same pixel could be reduced.

Another way to minimize the occurrence of this situation consists in using a more expensive and precise camera in order to increase the measurement resolution. The technical specifications of the camera used for this experimentation can be read in the Appendix A.1.

Improving the points acquisition

The only feasible solution in order to reduce the inaccuracies due to the IR LED (Section 7.4) is to equip smaller and more concentrated light sources on the arms. In this way, it would be more reliable the computation (Chapter 6) of the central point of the projected light.

Choice of better 3D points

In order to increase the accuracy of the alternative method, some attempts about the configuration of the 3D points can be done.

The alternative method finds the optical center as intersection of four straight lines (Section 5.1). For each line, four points with the same projections are taken. The problems mentioned above contribute to create inaccuracies, which, can be reduced in three ways:

- increasing the number of points per line, so that the least squares method used to find each line provides more accurate results
- increasing the number of lines, so that the optical center as the intersection of these lines is more reliable
- increasing the distance between the points of each line, so that the errors would decrease.

A further improvement could be obtained, according to observations in Section 7.3, choosing the positions on which the robots are moved, so that the projections are distributed all over the image. This could allow to reduce the mistakes in phase of the calibration of the camera with respect to the robots.

Future works

The interface described in Chapter 6 allows detecting the points on which an IR LED is moved, but the robot arm which drives the LED is remote-controlled by means of another interface (Fig. 6.2). In order to move the robot, the operator has to insert manually the coordinates or the angles to be sent to the control.

The acquisition of points is based on a start-stop procedure which involves the operator click more times two buttons in order to freeze the robot each time it is in the right position.

A possible improvement in order to make the process more efficient and faster is the automatization of the points capture. The interface which acquire the frames could be coupled with a TCP/IP socket in order to communicate to the robot a set of predefined positions. The positions (X, Y, Z) with their projection (u, v) could be automatically saved in a text file when the robots returns a confirmation message.

BIBLIOGRAPHY

[1] Joaquim Salvi, Xavier Armangue, Joan Batlle, *A comparative review of camera calibrating methods with accuracy evaluation*

[2] Roger Y. Tsai, *“A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses”*. IEEE journal of robotics and automation, vol. Ra-3, no. 4, august 1987.

[3] Yannick Morvan, *Acquisition, Compression and Rendering of Depth and Texture for Multi-View Video, Ch. 2* (Projective Geometry), June 9, 2009

[4] Gary Bradski, Adrian Kaehler, *“Learning OpenCV”*, O’Reilly, Chapter 11, pp. 370-403.

[5] Bouguet, J.-Y, *Camera Calibration Toolbox for Matlab*. Retrieved 2013, from http://www.vision.caltech.edu/bouguetj/calib_doc/, July 9, 2010.

[6] David Balihar, *What is a pinhole camera?* Retrieved 2013, from http://www.pinhole.cz/en/pinholecameras/whatis.html, 2001-2013

APPENDIX

A.1. Camera specifications

Model No.	WV-CP470
Pick-up Device	771(H) x 492(V) pixels, Interline Transfer CCD
Scanning Area	4.8(H) x 3.6(V) mm (Equivalent to scanning area of 1/3" pick-up tube)
Scanning	525 lines/60 fields/30 frames Horizontal: 15.734 kHz Vertical: 59.94 Hz
Synchronization	Internal, Line-locked, External (VS/VBS) or Multiplexed Vertical Drive (VD2) selectable
Video Output	1.0V[p-p] NTSC composite 75Ω/BNC connector
Horizontal Resolution	480 lines (C/L), 570 lines (B/W)
Signal-to-Noise Ratio	50dB (Equivalent to AGC Off, weight On)
Dynamic Range	46dB
Minimum Illumination	0.8 lx (0.08 fc) at F1.4 (C/L), 0.1 lx (0.01 fc) at F1.4 (B/W)
Gain Control	ON (DNR-H), ON (DNR-L) or OFF (SET UP MENU) selectable
White Balance	ATW1, ATW2 or AWC (SET UP MENU) selectable
Aperture	Set Variable (SET UP MENU)
Electronic Light Control	Equivalent to continuous variable shutter speeds between 1/60s and 1/10,000s
Super Dynamic II	ON or OFF (SET UP MENU) selectable
Electronic Shutter Speed	1/60(OFF), 1/100, 1/250, 1/500, 1/1,000, 1/2,000, 1/4,000, 1/10,000s selectable
Lens Mount	CS-mount (supplied with C-mount adapter)
ALC Lens	DC or Video selectable
Ambient Operating Temperature	–10°C ~ +50°C (14°F ~ 122°F)
Ambient Operating Humidity	Less than 90%
Power Source and Power Consumption	120V AC 60Hz, 4.8W
Dimensions	70(W) x 65(H) x 128(D) mm [2-3/4"(W) x 2-9/16"(H) x 5-1/32"(D)]
Weight (approx.)	460 g (1.01 lbs.) (without power cord)

Table A.1 - Specifications of the employed video camera